RECOGNITION OF ARABIC ONLINE HANDWRITTEN TEXT USING SYNTACTICAL TECHNIQUES

BY

Khaldoun Mohammad Khaled Hassan Halawani

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

INFORMATION AND COMPUTER SCIENCE

April 2013

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN- 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **KHALDOUN MOHAMMAD KHALED HASSAN HALAWANI** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE.**

Dr. SABRI MAHMOUD
(Advisor)

Dr. Adel Ahmed
Department Chairman

Dr. RADWAN ABDEL-AAL
(Member)

Dr. Salam A. Zummo
Dean of Graduate Studies

Dr. WASFI AL-KHATIB
(Member)

30/6/13

Date

2013

To my mother,

and the memory of my father

ACKNOWLEDGMENTS

All praises due to Allah who has permitted us and has given us the ability to complete this research work.

Thanks to King Fahd University of Petroleum and Minerals and the Kingdome of Saudi Arabia for giving me this research opportunity and providing me with all what I need.

I must thank my advisor Prof. Sabri Mahmoud for his guidance and great advice and support during my MS study. I'm also grateful for his quick response and help whenever a problem comes up. Appreciations are also to my academic committee members Prof. Radwan Abdel-Aal and Dr. Wasfi Al-Khatib for their significant suggestions and support.

I also thank all of my colleagues at KFUPM who helped my directly or indirectly to do this work. Namely many thanks due to Dr. Mohammad Tanvir Parvez for supporting me with his previous work.

I'm extremely grateful to my family for their encouragement, support and prayers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Full Name : Khaldoun Mohammad Khaled Hassan Halawani

Thesis Title : Recognition of Arabic Online Handwritten Text using syntactical techniques

Major Field : Information and Computer Sciences

Date of Degree : April 2013

In this thesis, we have addressed Arabic online handwriting recognition using syntactical techniques. The applications of automatic Arabic online text recognitions are several. Data input in stylus supported devices is the main application, writer identification and forms processing are other applications. Recently large number of research work is published in this field. Latin and Chinese automatic online handwriting recognition are much more advanced than other languages including Arabic. This may be attributed to the cursive nature of Arabic writing, ligatures and the lack of comprehensive benchmarking databases and resources.

Previous research on online Arabic handwriting recognition focused on the recognition of isolated characters, numerals and words. Different methodologies have been reported in the segmentation and recognition phases. Statistical techniques are used to segment and recognize cursive text. Classifiers such as Hidden Markov Models (HMM) and Neural Networks are used in the recognition phase. On the other hand, structural methods are not widely used in this field. Some systems applied a combination of classifiers that used both structural and statistical techniques.

In this work, we introduced new techniques for the different phases of online Arabic handwriting recognition. A rule based segmentation algorithm is presented. The algorithm aims to decompose the cursive online text into segmented letters. The algorithm implemented a set of rules to extract the Possible Segmentation Points (PSPs) of the online Text. These rules are easily updatable. In addition, another rule based algorithm to extract the attachments of the Arabic words is used. This algorithm effectively separates the main body of the online text from its attachments and additions. An improved fuzzy Arabic characters' modeling is introduced. This modeling technique utilizes the polygonal approximation of the Arabic letters. It applies turning functions to the letters' polygons. In the training phase the turning function is used to extract the character models. The generated fuzzy models are evaluated by comparing the similarity between the classes' models.

The presented work is applied and tested on different benchmarking and private databases. Acceptable recognition rates are achieved, given it is applied to unconstrained online handwritten text. This includes the combined segmentation and recognition phase. It is known that the segmentation errors increase the recognition errors nonlinearly.

These results motivate us and other researchers to put more efforts in this subject as there is big room for improvement. In addition, our new techniques can be extended and enhanced in different ways to obtain higher performance results.

ملخص الرسالة

الاسم: خلدون محمد خالد حسن الحلواني

عنوان الرسالة: التعرف الآلي على الكتابة اليدوية العربية الآنية بإستخدام خصائصها البنيوية

التخصص: علوم الحاسب الآلي

تاريخ الدرجة العلمية: نيسان 2013

قمنا في هذه الرسالة بدراسة التعرف الآني على الكتابة اليدوية العربية بإستخدام الصفات البنيوية. تتعدد تطبيقات أنظمة التعرف التلقائي على الكتابة العربية. أبرز هذه التطبيقات هي عملية إدخال البيانات من خلال الأجهزة المدعومة بالأقلام الإلكترونية. إضافة إلى ذلك يعتبر التعرف على الكاتب و معالجة النماذج من التطبيقات الأخرى. تم نشر عدد كبير من الأبحاث في هذا المجال مؤخرا. تتقدم اللغات اللآتينية و اللغة الصينية على اللغات الأخرى و من ضمنها العربية في هذا المجال. قد يعود السبب في ذلك إلى طبيعة الكتابة المتصلة و كتابة الحروف المتراكبة في اللغة العربية إضافة إلى نقص وجود مصادر و قواعد بيانات معيارية شاملة.

ركزت الأبحاث السابقة في مجال التعرف على الكتابة اليدوية العربية على الحروف المنفصلة، و الأرقام بالإضافة إلى الكلمات المنفصلة. حيث تم إستخدام العديد من المنهجيات لتنفيذ مراحل تقطيع الكلمات و التعرف عليها. استعملت الطرق الإحصائية لتنفيذ عمليات التقطيع والتعرف على النصوص المتصلة. من الأمثلة على ذلك إستخدام نماذج ماركوف الخفية

(Hidden Markov Models), والشبكات العصبية (Neural Networks). من جهة أخرى لم يتم إستخدام المنهجيات التي تعتمد على الصفات البنيوية في هذا المجال. لكن قامت بعض الأنظمة و الأبحاث بدمج الأساليب الإحصائية مع الأساليب البنائية.

نقدم في هذه الرسالة أساليب و طرق جديدة لتنفيذ المراحل المختلفة في عملية التعرف على الكتابة اليدوية العربية. حيث تم إستحداث خوارزمية مبنية على القواعد لتقوم بعملية تقطيع النصوص المتصلة. تهدف الخوارزمية إلى تقسيم النصوص المتصلة المكتوبة بشكل آني إلى حروف منفصلة. تقوم الخوارزمية بإستخدام مجموعة من القواعد من أجل إستخراج نقاط القطع المحتملة -(Possible Segmentation Points (PSP- في النص الآني. تتميز هذه القواعد بسهولة تحديثها وتعديلها. إضافة إلى ذلك، تم إستحداث خوارزمية أخرى مبنية على القواعد للقيام بعملية فصل الحروف العربية عن تشكيلاتها و النقاط المرتبطة بها. كما طورنا في هذه الرسالة اسلوب محسن لتمثيل الحروف العربية بشكل ضبابي. يقوم هذا الأسلوب على الإستفادة من المضلعات التقريبية لأشكال الحروف وزوايا هذه المضلعات لتمثيل الحروف. تم إستخدام هذه النماذج للقيام بعملية تقييم التشابه بين نماذج الحروف.

وقد تم تطبيق التقنيات المقترحة على بعض قواعد البيانات المتوفرة القياسية والخاصة. وحصلنا على نتائج مقبولة مع الأخذ بعين الإعتبار أنه تم إختبارها على نصوص عربية يدوية غير مقيدة. يتضمن ذلك كلا من عمليات التقطيع والتعرف على الكلمات. ومن المعلوم أن الأخطاء الناجمة عن التقطيع تؤدي إلى زيادة كبيرة في أخطاء التعرف على الكلمات.

إن النتائج التي توصلنا إليها في هذه الرسالة تحفزنا وغيرنا من الباحثين على بذل جهود أكبر و ذلك لوجود مساحة كبيرة من التطوير و تحسين الأداء. إضافة إلى إمكانية تطوير هذه التقنيات وتطبيقها بطرق مختلفة للحصول على نتائج أعلى.

CHAPTER 1

INTRODUCTION

Automated online handwriting recognition can offer smooth and improved way to interact with computers and other devices. Handwriting can be one of the most common interaction techniques in such devices. In addition, the increasing number and diversity of hand held devices and tablets, with touch screens, increased the need for reliable online handwriting text recognition systems. The field of handwriting recognition started with the objective of recognizing separated characters and digits numbers [1], [2] , and [3]. Then this objective was extended to include the recognition of complete words and paragraphs [4], [5]. This research area has been intensely researched for English/Latin languages. High recognition rates are published for other languages such as Latin and Chinese languages. Arabic handwriting recognition has two main difficulties. The first one, which is common to other languages, is that handwriting of one person may have different style than another person. Hence, statistical techniques may not be very helpful. The other difficulty for Arabic language text recognition is that it is cursive. The challenge here is that we need to segment a letter in order to recognize it, or to model it in a way that can recognize the letters without segmenting them based on their shape and the way letters are written. This thesis mainly aims to tackle these two problems.

1.1. Motivation

The importance of this research arises from the importance of Arabic language. Arabic is the mother tongue of more than 310 million native speakers [6], and is used by more than 1.5 billion Muslims across the world [7]. Arabic written script uses a standard form usually called “Modern Standard Arabic (MSA)” which is used by native Arabic speakers. Moreover, Arabic language script is used by more than one language, such as Urdu, Persian, Malay and Kurdish. This shows the huge benefit of an Arabic scripts automated recognition system. On the other hand, the recent advancements in the area of hardware devices have produced a wide spectrum of devices that enables online handwriting systems. PDA’s, Tablet Computers, Smart Phones, touch screens, electronic notepads, media tablets and many other technologies increase the need for reliable online handwritten text recognition. Our literature review shows that Arabic language lacks online Arabic text recognition systems. Considering these factors; intensive research and development is needed in the field of online Arabic handwritten text recognition. The main aim of this thesis is to research this area and implement a prototype for online Arabic text recognition.

1.2. Problem Definition

Handwritten text recognition can be classified into offline and online text recognition. Offline systems aim to recognize texts represented in image format. Mainly the texts are written on paper or using stylus supported devices, then these writings are stored as images. Online handwriting is represented as a sequence of (x, y) coordinates that represent each point on the handwriting. The usage of online data gives additional information about the content of the handwriting. This additional information includes

the time of writing by storing the points as they are written in sequence. The time each segment/ stroke is written is an additional feature about that segment or stroke. Using this information gives an indication about the time of each part of writing which helps in enhancing recognition. In this work, we are addressing online text recognition. An online database and a classifier that utilizes this online information are implemented.

Online handwriting recognition has many applications, such as text input for stylus supported devices, writer identification, electronic signature verification and many other applications. Our main objective is to address online Arabic text recognition on the word/ sub word levels. This recognition may be extended to the line or paragraph levels. The syntactical (structural) features are used to describe the shape and structure of the text. The usage of these features gives us an indication of the way the text has been written. An online Arabic text recognition prototype is developed. Different special and standard online Arabic text databases [31] are used. Those databases are used to train and test the system prototype. The performance of the prototype is evaluated. In general, the system prototype takes an online handwritten text as input; the prototype builds models in the training phase and classifies the text in the testing phase. The output will be a digitally recognized text.

1.3. Contributions of the Thesis

In this section we summarize the contributions of this thesis to online Arabic handwriting recognition using structural attributes.

- **Letters attachments:** A rule-based algorithm is implemented that is easy to update. These rules validate the structural properties of the strokes in a data

sample. The strokes that match the rules are considered as attachment. New rules can be added and old ones can be modified or removed easily. The algorithm links each attachment to its most probable parent.

- **Possible Segmentation Points (PSP):** The proposed algorithm is rule based that uses a set of rules to identify the best PSP of the online text. The suggested rules are derived based on analyzing the connectivity of Arabic text and on the general features of text. The segmentation algorithm is evaluated manually and semi-automatically using our database and ADAB database.
- **Fuzzy structural models of Arabic characters:** The thesis proposed enhanced fuzzy models of Arabic letters. Polygonal approximations of Arabic characters are generated. A turning function model is generated from the polygonal approximation models. The general model of a letter is obtained by accumulating the set of samples of characters. This automatic model is used to generate a fuzzy model of Arabic characters with fuzzy directions and lengths. This model is considered as an enhancement to the previous work of Parvez [8] in the following aspects:
    - We used fuzzy directions and lengths while only fuzzy directions were used in Parvez work.
    - We used adaptive fuzzy margins at the different parts of each model while fixed margins were used in Parvez work [8].
    - The models are generated automatically from the training samples.

    This makes the model more robust and representative of the different writing styles of writers.

- **Prototype for online Arabic handwriting text recognition.** A prototype is implemented that utilizes all the techniques proposed in the thesis which provide multiple hypotheses resulting in lists of possible letters. The lists are used to generate multiple hypothesized words. These words can be used to improve recognition rates by using a language model or a dictionary.

The rest of the thesis is organized as follows. In Chapter 2 we present a literature review of research on online Arabic handwriting recognition. Chapter 3 discusses the characteristics of Arabic language script, and the organization of Arabic letters. Chapter 4 presents the possible segmentation points (PSP) detection and segmentation of text. It also details the experimental results of this phase. Chapter 5 discusses Arabic letters' modeling. It presents the proposed fuzzy models and its analyses. The text developed recognition prototype is discussed in details in Chapter 6. Finally Chapter 7 concludes this thesis and summarizes the outcomes and future work.

CHAPTER 2

LITERATURE REVIEW

Many works are published in the field of online handwriting text recognition. Most of that work addresses English/Latin languages. The literature review showed good results for English/Latin. For Arabic language we can find comparatively limited research works on online handwriting recognition. Most of such work focuses on the recognition of isolated characters and digits. Some techniques addressed cursive text recognition. Some techniques used statistical features, and others used structural features. In this section we review the state of the art on online text recognition.

A comprehensive survey of off-line and online handwriting recognition researches for Ethiopic scripts is presented in [9]. Online handwriting recognition technologies that focus on the real applications of online handwriting in Nepalese scripts are surveyed in [4]. In this survey an online character recognition framework is proposed and the most famous technologies are discussed.

Different segmentation techniques of Arabic online handwritings are discussed in [10]. It emphasized the importance and the need for online Arabic recognition systems. In addition, different segmentation techniques of the online Arabic scripts are compared and their strengths and weakness are stated.

Arabic Handwritten character recognition is addressed in [11]. Their system was divided into two parts: preprocessing and recognition. For preprocessing an algorithm, which

produces skeletons that consist of the structure of the components of the characters, was developed. Then the character skeletons are converted to a tree structure for recognition. In the recognition phase, flexible models for each character are produced. These models are fuzzy constrained character graph models. Handwritings of four Arabic writers were used, which is very limited.

An online recognition system for handwritten Arabic characters is presented in [2]. A graphics tablet was used to input the words and then words were segmented into strokes. The length and direction of each segment were found to help with categorization. The system is trained in the training phase. In the recognition phase, the attributes of each stroke are found then rules are used to find the best matching of the trained models. The authors considered each style of a character as a different model.

A non-parametric polygonal approximation is introduced in [12]. The algorithm may be used for digital planar curves. Firstly, the algorithm identifies cut-points of the contour. By applying an optimization function, the best fitting polygonal shapes are then found using the cut-points. Suppression techniques for addressing noise were used. The performance of the algorithm was tested on a large set of Arabic handwritten characters and MPEG7 CE Shape-1 Part B database. In this thesis we adapted this algorithm to Arabic online text recognition.

An efficient structural approach for recognizing online handwritten digits is proposed in [13]. The slope of the line segment is estimated using the X and Y coordinates of the input. The primitives are identified based on the changing signs of the slope. From these primitives, they determine the specific grammar that the digit belongs to. A Finite

Transition Network that contains grammars for all the digits was used to match the given primitive with the stored ones. The authors used 3000 digits written by 100 persons, each digit is written three times. The reported recognition rate is 95% on the test data.

A simple and robust structural approach for recognizing online handwriting alphanumeric characters was presented in [14]. Freeman's chain code is used to represent the directional information of the curves that compose letters and digits. They introduced a specific grammar for recognizing the primitives of the entered characters. In the recognition process, they extracted primitive structural features from the sequence of points captured by a digitizer. A flexible structural matching method is used to recognize the characters. Recognition rate of 97.4% on the test set (i.e. digits, uppercase, and lowercase letters) are reported.

Template-based writer-independent online character recognition was proposed in [15]. In this work, the writing is represented as a set of strokes that form the shape of the letter. Each stroke is represented by a sequence of (x, y) coordinates. String matching is used as a distance measure to find the distance between the character pattern pairs. An iterative elastic matching is used in order to match unknown characters against known characters. A total of 4085 uppercase and lowercase characters and digits were used to test the system performance using Nearest-Neighbor classifier. An average recognition rate of 86.75%, with 2 look ahead depth is reported. A rejection option was added to enhance the recognition rate. Some cases were improved by using the rejection criteria.

The structural features of the online Arabic handwritten characters are used in [13] in order to classify characters using a decision tree. They identified a set of structural

features by tracing the shape of the written character represented by sequence of (x, y) coordinates. Then they used the number of segments of each character, the existence of loops/ cross points, the existence and shape of sharp edges in the characters and several other structural features. A decision tree was built based on these features in order to find the class that represents the sample character. The reported recognition rate of the system was 75.3%. The recognition rate was improved to 85.31% by excluding a small set of characters "ج،ح،خ،ع،غ" that have similar structure according to their features.

Kherallah et al. [16], [17] presented the idea of combining the kinematics and geometry in handwriting trajectory modeling. These studies estimated the strokes number based only on the velocity signal extreme. They did not consider the overlap of Beta signal or the inflection points which present a key role for the strokes number determination. In [16] the modeling system was based on Beta-circular approach. In their later work [18], the authors used a method of the handwritten trajectory modeling based on elliptic and beta representation of online digits is presented. A classifier consisting of the Multi-Layers Perception Neural Networks (MLPNN) was developed in a fuzzy concept. The training process of the recognition system was based on an association of the Self Organization Maps (SOM) with Fuzzy K-Nearest Neighbor Algorithms (FKNNA). To test the performance, the authors built 30,000 Arabic digits. The reported global recognition rate was about 95.08%.

Assaleh et al. proposed an online video-based approach to handwritten Arabic alphabet recognition [19]. The motion information of the hand movement was projected onto two static accumulated difference images according to the motion directionality. The temporal analysis was followed by two-dimensional discrete cosine transform and Zonal

coding or Radon transformation and low pass filtering. The resulting time-independent feature vectors were then classified by K-Nearest Neighbor (KNN). The authors further enhanced the recognition by using super classes where similar classes are grouped together for the purpose of multi resolution classification. A recognition rate of 99% on a database comprising videos of 28 isolated Arabic letters (each letter is repeated eight times by two different users) is reported.

Several statistical features were used in AraPen system [20]. These include: (x, y) co-ordinates, tangent angles, winding values (algebraic sum of direction changes) and aspect ratio. These features were utilized in a two-level classification system. In the first level, Dynamic Time Warping (DTW) is used. The best three candidates selected in the first level are then further classified using neural network. In AraPen, cursive words are recognized by explicitly segmenting the word. This segmentation is based on locating horizontal strokes. Segmentation points are marked while the user writes on the tablet and updated adaptively.

Sternby et al. [22] presented a new template matching scheme and applied it to the recognition of cursive Arabic script. The new system was tested on a dataset consisting of 40 persons entering words from a list of 66 Arabic words (a total of 1,578 samples) and a collection of the isolated forms of single characters. The authors reported character recognition rates of 94.8% and word recognition rates of about 92%.

Researchers have utilized different classifiers for the recognition purpose. These classification algorithms include template matching [21], [22] decision trees [23], [2], [24], fuzzy logic reasoning [25], [26], [27], neural network mapping [28], dynamic

programming for cursive word recognition [1], and Hidden Markov modeling (HMM) [29] , [30], [31]. Combinations of different classifiers to improve recognition have also been investigated in [26], [28].

CHAPTER 3

ARABIC CHARACTERS' MODELING

In this chapter, we present the different concepts related to Arabic writing. The main characteristics of the written Arabic scripts are addressed, and the utilization of these features to model Arabic letters is presented. We described the shapes of the various letters and how we can gather these letters into classes. This chapter also discusses the polygonal approximation of online text. It shows the benefits of using such representation in both the PSP detection phase, described in Chapter 4, and in the fuzzy modeling phase, described in Chapter 5.

3.1. Characteristics of Arabic Text

Arabic written script is composed of 28 letters (called alphabet 'الأبجدية'). Each letter has different shapes and structure. Arabic is written from right to left. Several letters of the 28 set have similar shapes where dots and other secondary strokes are used to discriminate between these letters. The letters (ب، ت، ث) have the same shape, but with different number of dots. Additionally some letters have zigzag shape stroke, called hamza shape (ء), such as (ك، ئ). Sixteen letters have one, two or three dots in the alphabets. Table 1 shows those letters.

Table 1: Letters with attachments

ب	ت	ث	ج
خ	ذ	ز	ش
ض	ظ	غ	ف
ق	ك	ن	ي

The locations of these secondaries differ from one letter to another. It maybe located at the top of some letters like (أ ،ف ،ن ،ت), or under the letter (إ، ي، ب), or in the middle of the letter (ك، ج).

Arabic writing is characterized by its cursive nature in printed and handwritten texts. Most of the letters in Arabic has more than one form of writing, depending on its position on the word. The letter might take one of four shapes, for instance, the independent form of the letter is (ب), in addition to the beginning form (بـ), the middle form (ـبـ) and the end form (ـب). Figure 3.1 shows 4 different Arabic words where the letter (ع) appears in its four different shapes based on its position of the word.

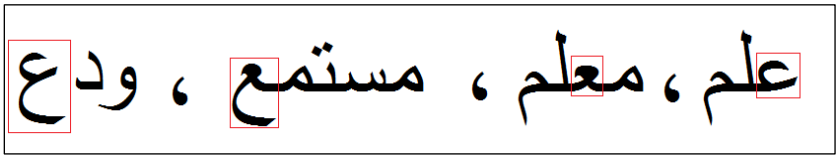


Figure 3.1: The letter (ع) in the beginning, middle, end of the word and isolated

There are no fixed dimensions of Arabic letters, such as fixed width or height, but it differs based on the writing style and the position of the letter in the word. Table 2 shows the full set of Arabic letters with the different forms of each letter, based on their position in the word.

Table 2: The Arabic alphabets

Contextual forms			
End	Middle	Beginning	Isolated
ـا	ـا	ا	ا
ـب	ـبـ	بـ	ب
ـت	ـتـ	تـ	ت
ـث	ـثـ	ثـ	ث
ـج	ـجـ	جـ	ج
ـح	ـحـ	حـ	ح
ـخ	ـخـ	خـ	خ
ـد	ـد	د	د
ـذ	ـذ	ذ	ذ
ـر	ـر	ر	ر
ـز	ـز	ز	ز
ـس	ـسـ	سـ	س
ـش	ـشـ	شـ	ش
ـص	ـصـ	صـ	ص
ـض	ـضـ	ضـ	ض
ـط	ـطـ	طـ	ط
ـظ	ـظـ	ظـ	ظ
ـع	ـعـ	عـ	ع
ـغ	ـغـ	غـ	غ
ـف	ـفـ	فـ	ف
ـق	ـقـ	قـ	ق
ـك	ـكـ	كـ	ك
ـل	ـلـ	لـ	ل
ـم	ـمـ	مـ	م
ـن	ـنـ	نـ	ن
ـه	ـهـ	هـ	ه
ـو	ـو	و	و
ـي	ـيـ	يـ	ي
ـئ	ـئـ	ئـ	ء

Our modeling of Arabic letters takes into consideration the characteristics of letters. The main characteristics reflect the number, shape and the structure of each letter attachments. The letters' models that describe these features are shown in Table 3. This table can be used in the system training and recognition phases. This table is used as a supplementary resource to enhance the recognition based on the main letter shapes and structures.

The first column in Table 3 is an index for the letter shape. The table lists the 108 shapes of the Arabic letters which represent the 28 letters in all possible forms/positions. The second column shows the actual letter shape in its printed form. Column (**C**) contains the model number of the letters where each set of similar shapes are represented by a unique model number. The model number form will be discussed in the following sections. Columns (**D**) indicates whether the letter has one or more attachments or not by indicating the number of these attachments.

Table 3: Arabic letters' features table

A	B	C	D	E	F	G	H	I
1	بـ	1	1	S	B	.	F	T
2	تـ	1	2	S	T	-	F	T
3	ثـ	1	3	M	T	^	F	T
4	جـ	2	1	S	B	.	T	T
5	حـ	2	0				T	T
6	خـ	2	1	S	T	.	T	T
7	سـ	3	0				F	T
8	شـ	3	3	M	T	^	F	T
9	صـ	4	0				T	T
10	ضـ	4	1	S	T	.	T	T
11	طـ	36	1	L	T	|	T	T
12	ظـ	36	2	L	T	|.	T	T
13	عـ	5	0	S			F	T
14	غـ	5	1	S	T	.	F	T
15	فـ	6	1	S	T	.	T	T
16	قـ	6	2	S	T	-	T	T
17	كـ	7	0				F	T
18	لـ	8	0				F	T
19	مـ	9	0				T	T
20	نـ	1	1	S	T	.	F	T
21	هـ	10	0				T	T
22	يـ	1	2	S	B	-	F	T
23	ـا	11	0				F	T
24	ـب	12	1	S	B	.	F	T
25	ـت	12	2	S	T	-	F	T
26	ـث	12	3	M	T	^	F	T
27	ـج	31	1	S	B	.	T	B
28	ـح	31	0				T	B
29	ـخ	31	1	S	T	.	T	B
30	ـد	14	0				F	T
31	ـذ	14	1	S	T	.	F	T
32	ـر	15	0				F	B
33	ـز	15	1	S	T	.	F	B
34	ـس	16	0				F	B
35	ـش	16	3	M	T	^	F	B
36	ـص	17	0				T	B
37	ـض	17	1	S	T	.	T	B
38	ـط	18	0	L	T	|	T	T
39	ـظ	18	1	L	T	|.	T	T
40	ـع	19	0				T	B
41	ـغ	19	1	S	T	.	T	B
42	ـف	20	1	S	T	.	T	T
43	ـق	20	2	S	T	-	T	T
44	ـك	22	1	L	T	ء	F	T
45	ـل	22	0				F	T
46	ـم	23	0				T	B
47	ـن	24	1	S	T	.	F	B

A	B	C	D	E	F	G	H	I
48	ـه	25	0				T	T
49	ـة	25	2	S	T	-	T	T
50	ـو	26	0				T	B
51	ـي	27	2	S	B	-	F	B
52	ـى	27	0				F	B
53	ـلا	28	1	L	T	|	F	T
54	ـا	29	0				F	T
55	ـب	30	1	S	B	.	F	T
56	ـت	30	2	S	T	-	F	T
57	ـث	30	3	M	T	^	F	T
58	ـج	31	1	S	B	.	T	B
59	ـح	31	0				T	B
60	ـخ	31	1	S	T	.	T	B
61	ـد	32	0				F	T
62	ـذ	32	1	S	T	.	F	T
63	ـر	33	0				F	B
64	ـز	33	1	S	T	.	F	B
65	ـس	34	0				F	B
66	ـش	34	3	M	T	^	F	B
67	ـص	35	0				T	B
68	ـض	35	1	S	T	.	T	B
69	ـط	36	1	L	T	|	T	T
70	ـظ	36	2	L	T	|.	T	T
71	ـع	37	0				F	B
72	ـغ	37	1	S	T	.	F	B
73	ـق	38	2	S	T	-	T	B
74	ـف	38	1	S	T	.	T	T
75	ـك	39	1	L	T	ء	F	T
76	ـل	39	0				F	T
77	ـم	40	0				T	B
78	ـن	41	1	S	T	.	F	T
79	ـه	10	0				T	T
80	ـة	42	2	S	T	-	T	T
81	ـة	42	2	S	T	-	T	T
82	ـو	43	0				T	B
83	ـي	27	2	S	B	-	F	B
84	ـى	27	0				F	B
85	ـلا	44	1	L	T	|	F	
86	ـء	45	0				F	T
87	ـبـ	46	1	S	B	.	F	T
88	ـتـ	46	2	S	T	-	F	T
89	ـثـ	46	3	M	T	^	F	T
90	ـجـ	2	1	S	B	.	T	T
91	ـحـ	2	0				T	T
92	ـخـ	2	1	S	T	.	T	T
93	ـسـ	48	0				F	T
94	ـشـ	48	3	M	T	^	F	T
95	ـصـ	49	0				T	T
96	ـضـ	49	1	S	T	.	T	T
97	ـطـ	47	1	L	T	|	T	T
98	ـظـ	47	2	L	T	|.	T	T
99	ـعـ	50	0				T	T

A	B	C	D	E	F	G	H	I
100	ـغـ	50	1	S	T	.	T	T
101	ـفـ	51	1	S	T	.	T	T
102	ـقـ	51	2	S	T	-	T	T
103	ـكـ	52	0				F	T
104	ـلـ	28	0				F	T
105	ـمـ	53	0				T	T
106	ـنـ	46	1	S	T	.	F	T
107	ـهـ	54	0				T	T
108	ـبـ	46	2	S	B	-	F	T

Attachment size is literally described in column (**E**), where the letter “L” means that the letter has large size attachment, letter “M” means medium size attachment and letter “S” means small size. The rest of columns are described later in this section. Table 4 illustrates the different shapes of the possible attachments in Arabic writing. Some attachments have different shapes. In addition the size and position of each attachment are shown.

Table 4: Letters' attachments' shapes, sizes and locations

Attachment	Shape	Location	Size
Single Dot		Above/Below	Small
Double Dots		Above/Below	Small
Triple Dots		Above	Medium
Hamza		Above/Below	Large
Alef		Above	Large
Shadda		Above	Large
Fatha/ Tanween Fatha		Above	Medium
Dama/ Tanween Dama		Above	Medium
Kasra/ Tanween Kasr		Above	Medium
Madda		Above	Small

We now describe Columns F through I of Table 3. The attachment position is also described literally in field (**F**) where “T” means top\above the letter and “B” means bellow/under the letter. Column (**G**) contains a simple representation of the attachment shape. For example the letter “ت” has three possible attachments with a dot (.) shape, and so on for the other letters.

Other main feature in the table in column (H) is describing the existence of loops in the letters. It holds the value (T or F) to indicate if the letter has any loops. Figure 3.2 illustrates an example of different Arabic letters with loops.

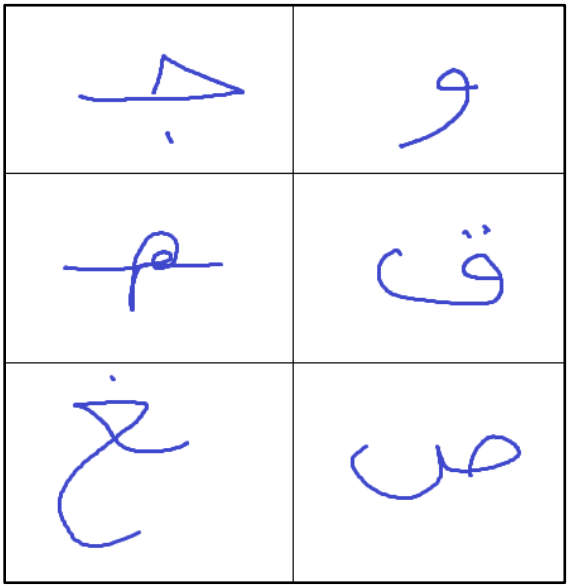


Figure 3.2: Different Arabic letters with loops

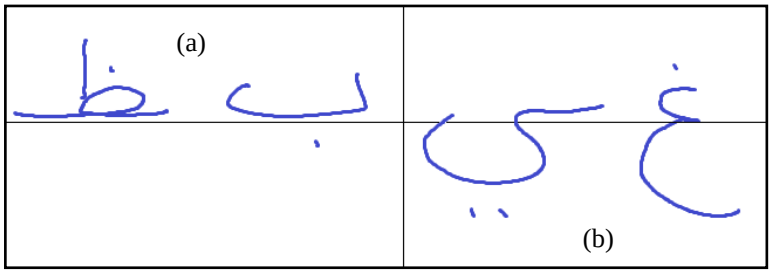


Figure 3.3: (a) Letters over the writing line. (b) Letters bellow the writing line

Column (**I**) identifies the position of the letter main body in relation to the writing line. Value "T" indicates that the letter is written above the writing line, while "B" indicates that it is written below the writing line. Figure 3.3 shows examples of different letters with different positions in relation to the writing line.

Table 3 of features can be considered as the ground truth that describes the attachments and the main characteristics of the Arabic letters' set.

3.2. Arabic Letters' Models

Arabic alphabet contains 28 different letters, and up to 108 different shapes/variations of these letters based on the position of each letter in the word. These variations increase the difficulty of recognizing the letters, due to the large number of possible letters in each recognition phase. To minimize these variations, this large set of possible letters has been grouped based on their basic shape/structure. For instance, letters (ج، ح، خ) have the same basic shape and hence have been given a unique number called *Model Number*. So those three letters are represented in a single class. Figure 3.4 shows examples of letters with similar shapes that can be grouped into a class.

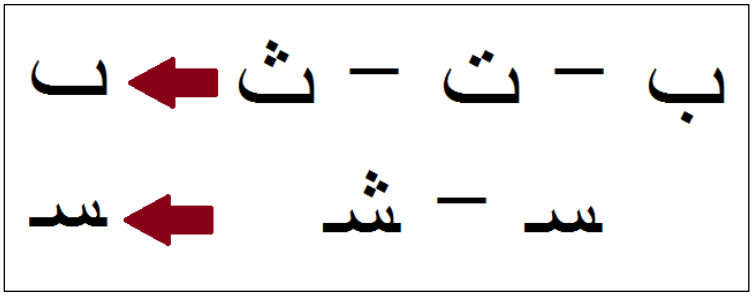


Figure 3.4: Examples of different letters' set in one class

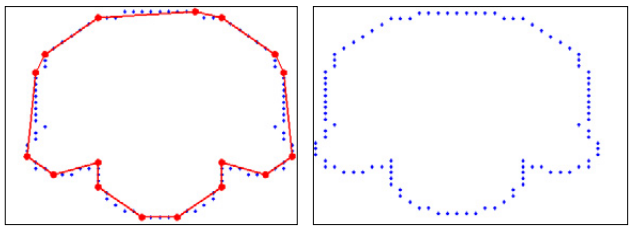
Table 3 shows the model number (Column-**C**) of the letters. The 108 different shapes of letters have been shorted into 54 models classes. Each model is composed of 2-5 different letters with the same basic shape. To generate these models, a database of letters has been built. The database contains different samples of different writers. The samples include all the 108 shapes of letters.

Reducing the number of classes can significantly improve the models in the training phase as the number of samples will increase for each model. In addition, it improves the recognition rates. The letters' attachments table is used to perform the intra class recognition by comparing the attachments' features of the sample against the expected letters in the resulting model (class).

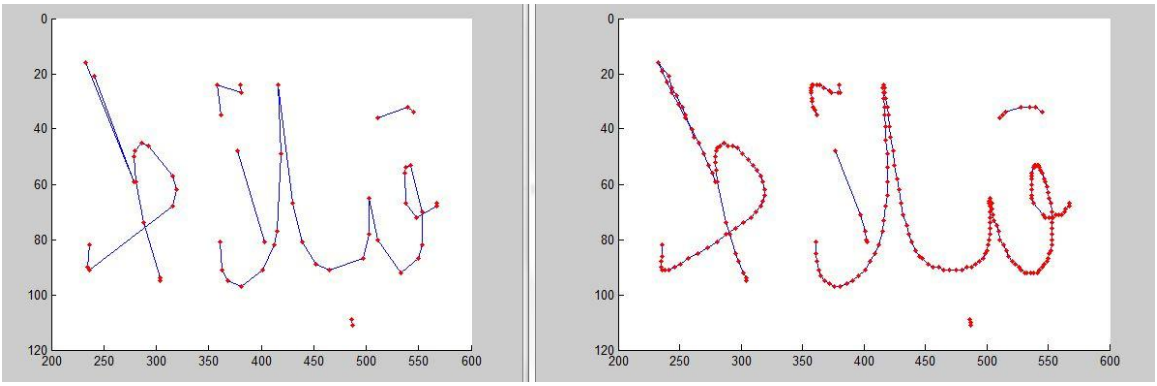
3.3. Polygonal Approximation

Polygonal approximation of online text is a pre-process for other phases. Polygonal approximation decreases the number of dominant points by removing less informative ones and leaving more essential points in the text. The aim of this process is to transform the text representation from series of (x, y) coordinates, into a series that describe the dominant/corner points in the online text. Using such an algorithm as a preprocessing phase has several advantages. The structure of the text is represented by the corner points of the online text. Polygonal approximation can be helpful also in removing the effect of spurious points and noisy parts in the online text. This makes the feature extraction phase more reliable, which will improve overall system performance.

Many polygonal approximation algorithms have been proposed in the literature [12] [32]. Each algorithm uses different methodology and has its own pros and cons. In our work, the “polygonal approximation of digital planer curve” in [12] is used. This algorithm uses a novel approach for adaptive polygonal approximation. It utilizes the advantage of both significance measure and suppression technique. In this technique, the online text is divided adaptively into segments and dominant points are found in the locality of a segment. The quality of polygonal approximation of that particular segment is used in choosing the dominant points. This approach gives the ability to handle different levels of details that are present in the online text. So it provides less number of dominant points in more straight portions of the segment and more dominant points in a higher level of detailed segmented. In addition, the algorithm proposes an efficient technique to remove the redundant dominant points called *Constrained Collinear-points Suppression (CCS)*. It effectively removes the noise of the online text by suppressing redundant points while preserving the shape of text. Figure 3.5 shows examples of the polygonal approximation algorithm applied on both digital planar images and online text. Our work utilizes the algorithm to transform an online text into a polygon. The right image in each sample is the input of the polygonal approximation and the left one is its output.



(a)



(b)

Figure 3.5: Examples of applying the polygonal approximation of (a) Digital planner curve [12] **(b) Online text**

CHAPTER 4

SEGMENTATION OF ONLINE TEXT

In this chapter we present our segmentation technique of Arabic online text. The segmentation process is an important phase in any online text recognition system. The input of the segmentation phase in our work is a set of strokes that represent the online text. The stroke contains a series of *x*- coordinates and *y*-coordinates that describe the shape of that stroke. Each of these strokes may be part of a letter, single letter or two merged letters, one of the secondaries or a whole word.

We initially find the Possible Segmentation Points (PSP). A PSP in online text is the point when the system recognizes a certain segment of the text. We apply polygonal approximation to strokes then we segment them into characters. Figure 4.1 shows an example of PSPs in an online text. PSPs are represented by small green circle around the intended points. In the following sections we address these phases in details.

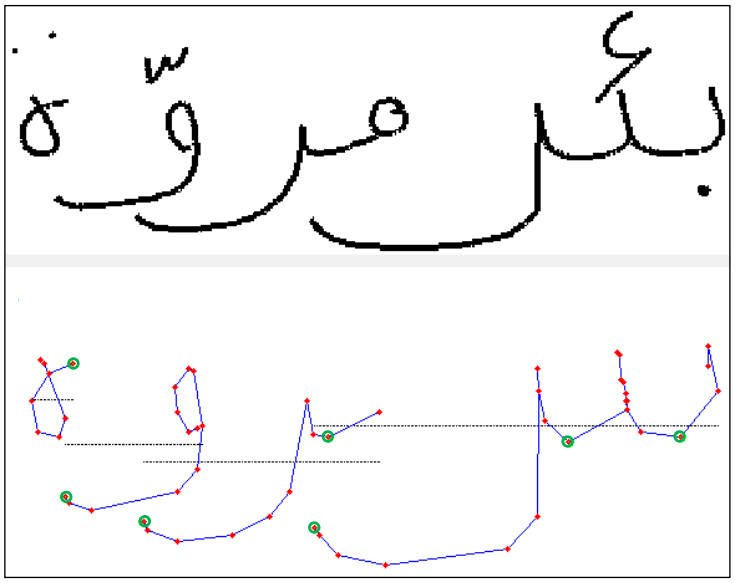


Figure 4.1: Text example after polygonal approximation and finding PSPs

4.1. Online Text Representation

The extraction of PSPs is done at the stroke level of the online text. Hierarchical online text representation is used in our work. The root of this representation is the online text, which can represent one word or more. The online text is represented by a set of *n* strokes; each stroke is *m* size series of (*x*, y) coordinates. By stroke we mean the writing from the pen down to pen up. Figure 4.2 shows the tree structure of the online text representation. The online text level is the raw data which represents the strokes. The segmentation phase decomposes and groups the strokes where each stroke is combined with its additions/attachments. The PSP's level represents the final output of segmentation.

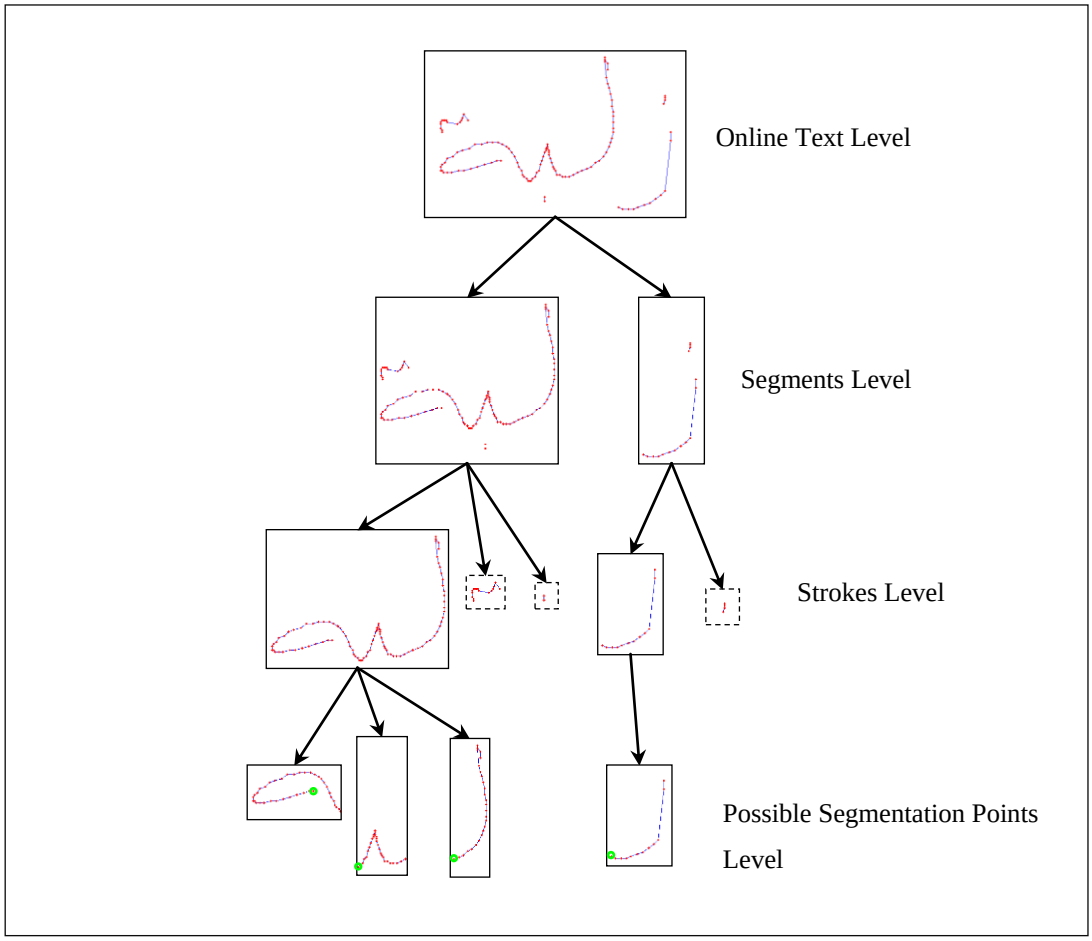


Figure 4.2: Online text hierarchal representation

4.2. Extraction of Letters' Attachments

One of the main challenges before the PSP extraction process is the extraction of each letter attachments. Excluding the attachments and keeping the main body of the text is helpful in further processing, since we don't need to find any segmentation points for those attachments. The process of extracting letters' attachments may be very difficult due to various reasons:

1-The different writing styles of attachments, so attachment can be written in different ways. Table 4 shows the different writing style of each attachment.

2- The misplaced attachments, where the writer writes the letter attachment in incorrect place relative to the letter. Figure 4.3 (a) shows examples of misplaced attachments.

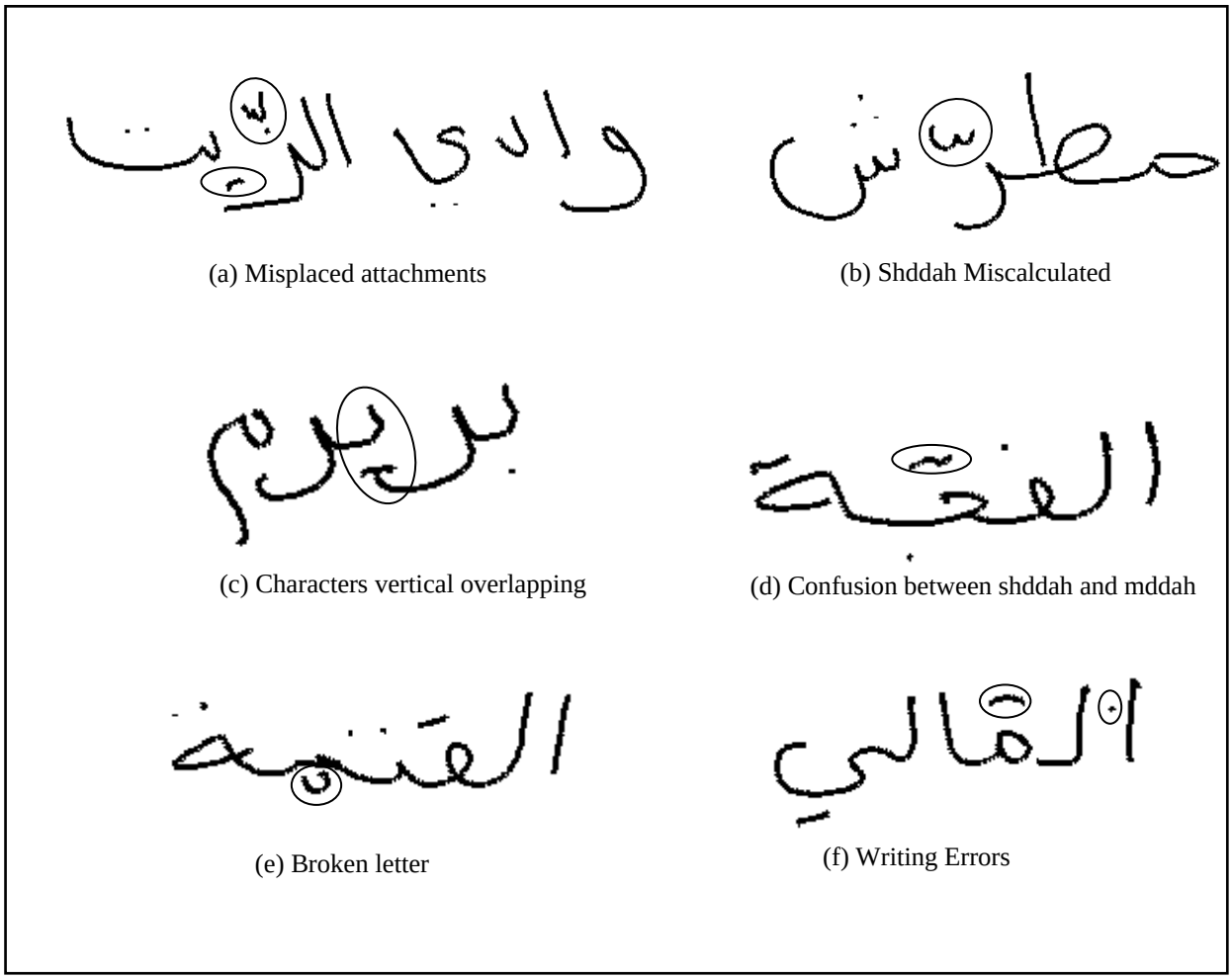
3- Misidentification of the attachments; as the attachment may be considered a part of the word main body. Figure 4.3 (b) shows an example of this case. Vertical overlapping between the attachment and the letters may result in the difficulty of extracting the attachment. In this case the attachment could be considered to belong to another character. Figure 4.3 (c) shows an example of vertical overlapping characters.

4- Confusion between the different types of attachments. This usually results because of the different writing styles. One attachment can be considered as another attachment. Figure 4.3 (d) shows an example of a shaddah (ّ) attachment that can be recognized also as maddah (**~**) or as double dots. The broken characters can be classified as an attachment instead of considering it a part of word. Figure 4.3 shows an example of letter (ـه) broken where the bottom part can be considered as an attachment of letter.

5- The existence of writing errors, where the writer may add extra strokes that can be considered as attachment.

6- The writer may use some attachment on the inappropriate letter. Figure 4.3 (f) shows an example of such cases.

The detection of words' attachments in our work follows a rule-base algorithm to identify whether a stroke is part of the text main body. This rule-based algorithm passes over each stroke in the online text and checks the possibility of that stroke to be a part of the main body.



(a) Misplaced attachments

(b) Shddah Miscalculated

(c) Characters vertical overlapping

(d) Confusion between shddah and mddah

(e) Broken letter

(f) Writing Errors

Figure 4.3: Different cases where extracting the attachments is considered a challenging process

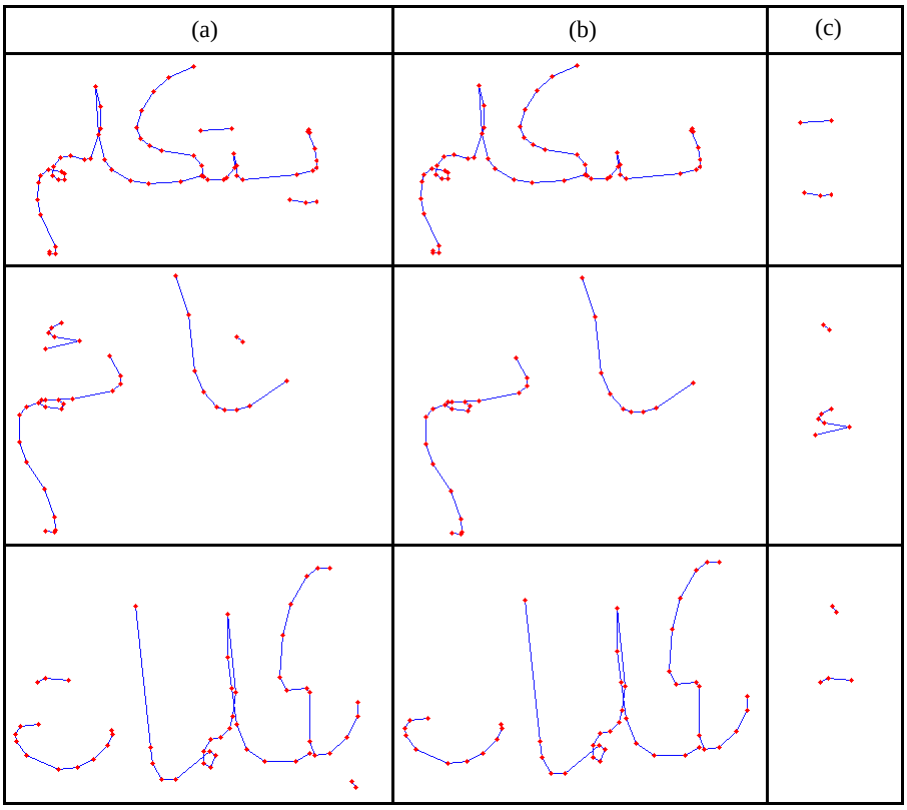


Figure 4.4: Different words before and after removing their attachments. (a): The original word (b) The word main body (c) The attachments of the word

To extract the attachments in each data sample we check each stroke in that sample against all other strokes. If the following three rules are satisfied then that stroke is considered initially as an attachment.

A stroke K is considered as an addition/attachment of other stroke (letter) S if:

Rule 1: The center of *x*-coordinates in stroke K is greater than the last *x* -coordinate in stroke S.

Rule 2: The center of *x*-coordinates in stroke K is less than the first *x* -coordinate in stroke S.

Merging rules 1 and 2, means that the stroke K is within the range of x-coordinates for the stroke S.

Rule 3: The size of stroke K is much less than the size of stroke S (less than half). Figure 4.5 shows an example satisfying these three rules.

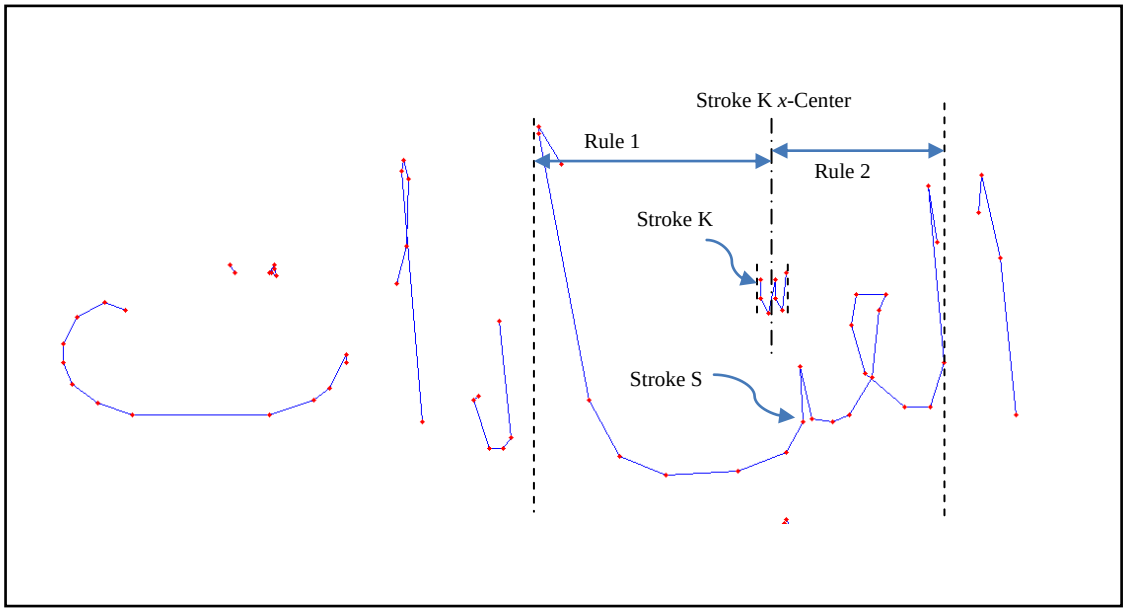


Figure 4.5: Occurrences of rule 1, 2 and 3 while extracting the attachments

Subsequently another 3 rules are executed. These rules check the strokes that are considered as part of the main body. The satisfaction of any of the rules results in considering the stroke to be an attachment.

Rule 4: The height of stroke K bounding box is much less than the overall text height.

Rule 5: The number of points in stroke K is small. Namely if stroke K has less than 3 vertices (points) it is considered to be an attachment. It rarely happens to have a letter with less than 3 points.

Rule 6: The distance between the center of stroke K and the text writing line is large; the acceptable distance is determined by multiplying a threshold by the high of the text writing line. Usually Arabic letters are written on the writing line; hence any stroke written away from it could be an attachment.

The second main stage in extracting the letters' attachments is to link each attachment to its parent letter. Each attachment is linked to the PSP of its letter. To do so we calculate the *x*-center of all segmented letters and attachments. Then for each attachment A we find the minimum horizontal distance between A's *x*-center and all segmented letters. The letter with the minimum horizontal distance is considered as the parent letter of attachment A. Figure 4.6 illustrates an example of the attachment of the letter Noon (ـنـ) in the word (جنتوره) after the process of PSP detection. As noted the distance d5, of letter (ـنـ), is the minimum among all other distances that represents other letters. So attachment A is linked with letter (ـنـ).

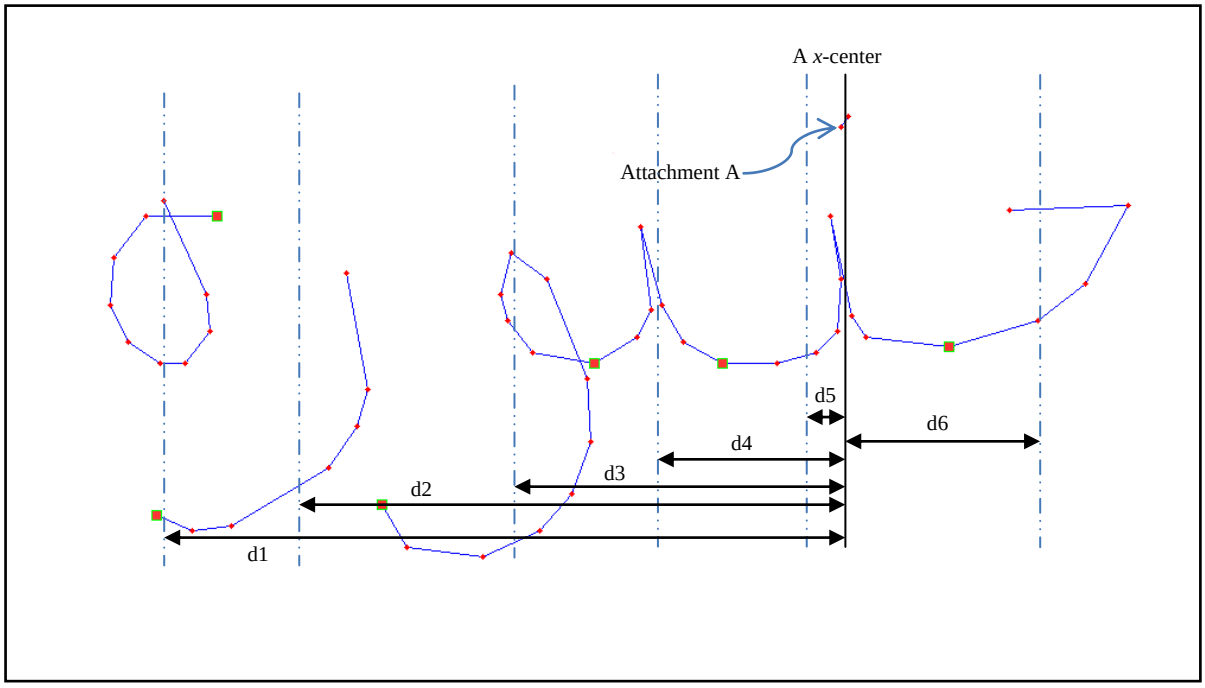


Figure 4.6: Attachments' parent matching based on horizontal distances

4.3. Detection of Possible Segmentation Points (PSP)

Finding the PSPs of the online text, (aka, segmentation of text) is done by another algorithm. The proposed algorithm is made easy to modify without the need for extensive modifications. A semi-automatic prototype is built to perform and evaluate the PSP detection process. A set of rules are applied to decide if each point can be considered a PSP. Another set of rules check if the selected points are not appropriate to be PSP, if so these points are removed from the PSPs list. Figure 4.7 shows the flowchart of this algorithm.

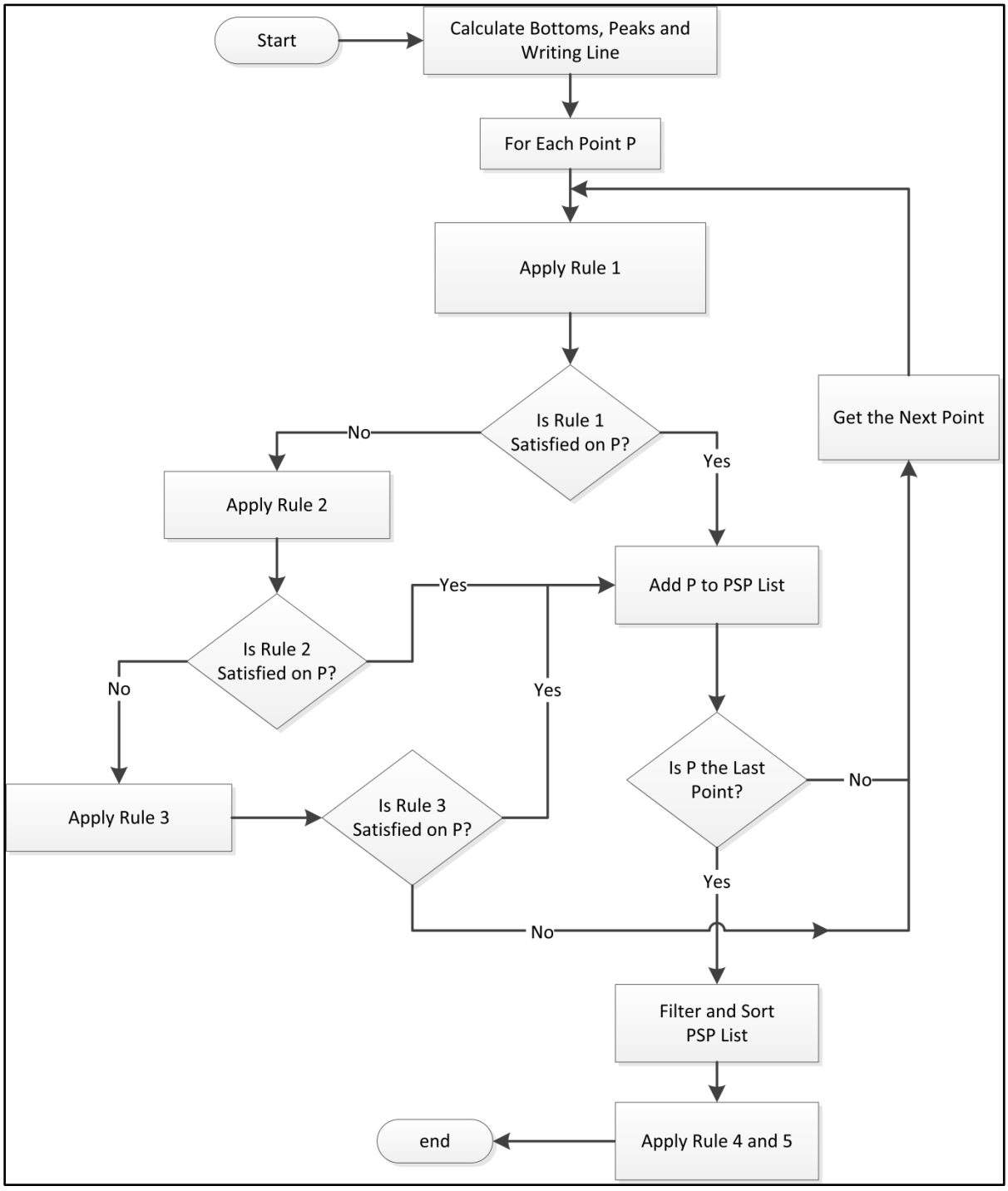


Figure 4.7: Decision points' finding algorithm

Let S (x_i, y_i) be the coordinates of the points in a stroke, where i=1, 2 …N. The following three rules are applied in order to find the PSP candidates.

Rule 1: A Point P (x_i, y_i) $\in$ S (x_i, y_i) is a candidate PSP point if one of the following four conditions is satisfied:

Condition 1: P (x_i, y_i) is lower than the previous P (x_{i-1}, y_{i-1}), i.e. $y_{i-1} > y_i$.

Figure 4.8 shows an example where P (x_i, y_i) having a smaller y value than its preceding point.

Condition 2: P (x_i, y_i) is lower than the next point in the stroke, i.e. $y_{i+1} > y_i$.

Figure 4.8 shows an example of the same P (x_i, y_i) that has a smaller y value than its subsequent point.

Condition 3: P (x_i, y_i) slopes from the right to left, or $x_{i-1} > x_i$. This means that the direction of writing for P (x_i, y_i) is from the right to left. Hence P (x_i,y_i)*:* $x_{i+1} < x_i < x_{i-1}$. Figure 4.8 shows an example of P (x_i, y_i).

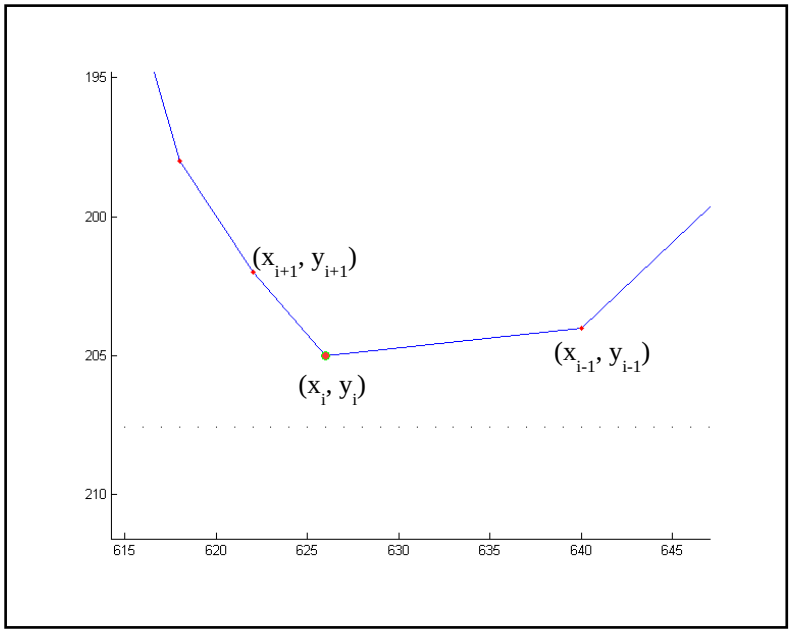


Figure 4.8: The occurrence of conditions 1, 2 and 3 (of rule 1) on point P (x_i,y_i)

Condition 4: The distance between P (x_i, y_i) and the writing line is less than a threshold (as it is shown in the formula).

$$|y_i - Writing\ Line| < Writing\ Line + threshold$$

Knowing that, "Writing Line" (aka, Y-writing line) represents the y-coordinate relative to the whole word coordinates. This makes the distance from the writing line correlated to the overall height of the text. Figure 4.9 shows P (x_i, y_i) as an example of this condition.

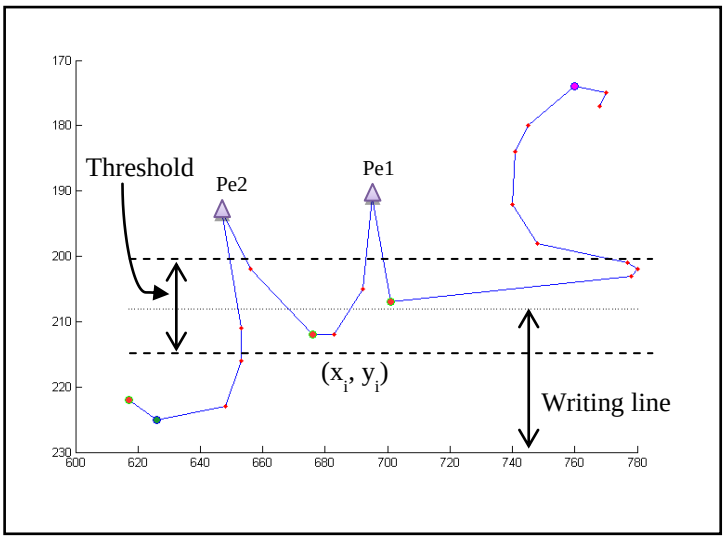


Figure 4.9: An example with satisfied rules 1 (4th condition) and rule 2 (1st condition)

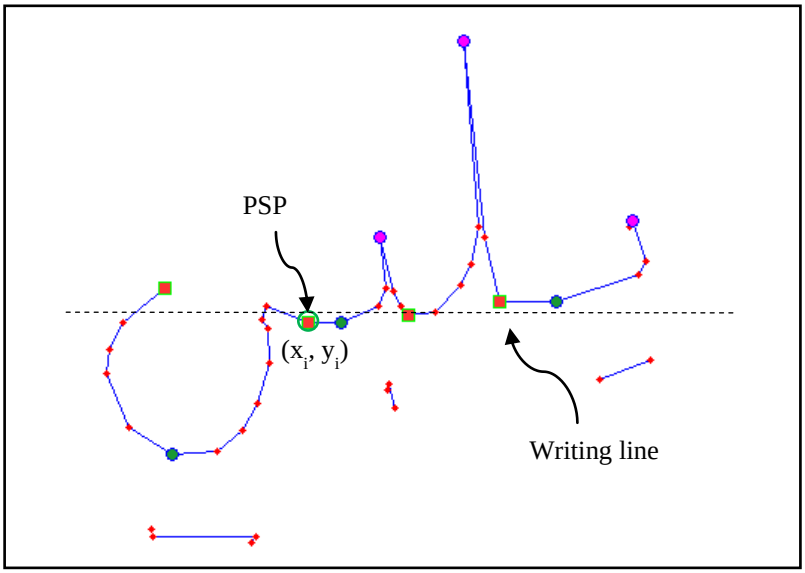


Figure 4.10: Example of satisfied rule2 (2nd condition)

Rule 2: A Point P (x_i, y_i) $\in$ S (x_i, y_i) is a candidate PSP point if one of the following three conditions is satisfied:

Condition 1: P (x_i, y_i) is a bottom point and it falls between two peaks. While writing an Arabic text, the movements from a top coordinate to a lower one, then moving up usually means that the writer started to write a new letter. Figure 4.9 shows the satisfaction of this condition. The peaks are marked with triangles. P (x_i, y_i) is a bottom point that falls between peaks P_{e1} and P_{e2}.

Condition 2: Most of the points subsequent to P (x_i, y_i) fall below the writing line is true. This means that the average values of the subsequent y values must be smaller than the writing line.

$$\frac{\sum_{v=i+1}^{n} y_v}{n-i} < Writing\ Line$$

Figure 4.10 shows an example of this case of the word (يلبي), the letter Yaa (ي) at the end of the word is segmented because most of its points are falling below the writing line. This means that the letter Yaa (ي) can't be segmented so we consider the P (x_i, y_i) as a PSP.

Condition 3: Any P (x_i, y_i) that falls before a loop. In most of Arabic letters with loops, the loop usually occurs at the beginning of the letter. Figure 4.11 shows the word (بكلمات) which contains a loop in the letter (ـمـ). This condition caused P (x_i, y_i) to be considered as a PSP.

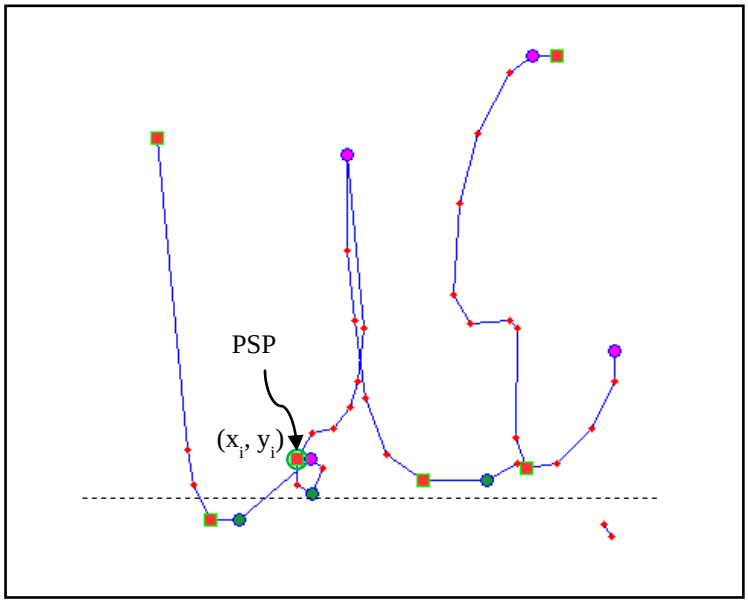


Figure 4.11: The points preceding a loop is a PSP

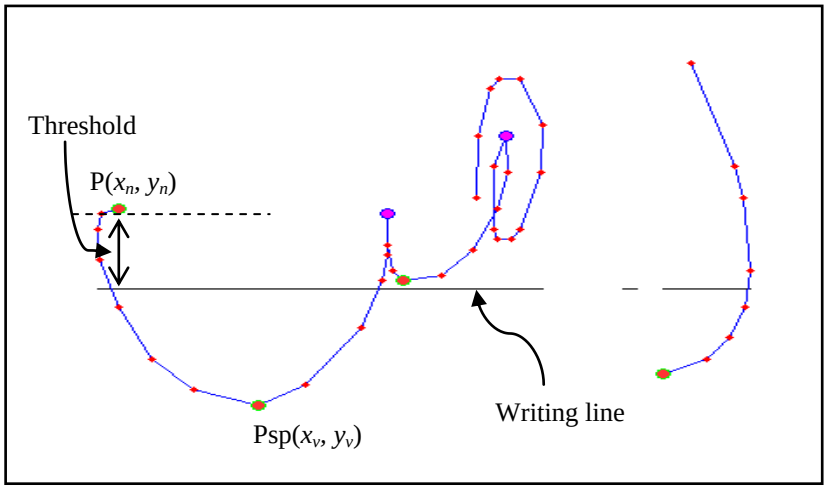


Figure 4.12: An example satisfying rule 3

Rule 3: Consider the point P (x_n, y_n), which represents the last point in each stroke as a decision point. This rule checks if any of the last 3 points before P is selected as a decision point then it is removed keeping P as PSP.

The following two rules are applied on the stoke points and the PSP list to further reduce it.

Rule 4: Consider P (x_n, y_n) as the last point in the stroke, and PSP (x_v, y_v) the penultimate PSP. Then if P is not a peak point or:

$$|y_n - Writing\ Line| < Writing\ Line + threshold \quad and \quad y_v \leq y_n$$

In this case the PSP (x_v, y_v) must be removed from the PSPs list. Figure 4.12 shows an example satisfying rule 4. The point PSP (x_v, y_v) is originally selected to be a PSP. The satisfaction of Rule 4 deletes this point from the PSPs list, as P (x_n, y_n) is close to the writing line

Rule 5: Consider PSP (x_v, y_v) as a candidate PSP. Remove this point if it falls far below the writing line:

$$|y_v| < Writing\ Line + threshold$$

In Arabic no letter starts under the writing line. This fact is utilized in Rule 5 to enhance the PSPs selection.

The PSP detection algorithm passes through each point in every stroke. Each stroke in the word will be checked independently. This results in a set of decision points for each stroke in the word. Figure 4.7 shows the PSP detection algorithm.

During PSP detection, some simple filtering logic is done to get the best minimum set of decision points. The first filter removes the duplicate decision points. Another filter is used to minimize the number of decision points by removing the second PSP in any two consecutive decision points. The reason is that almost impossible to have a letter represented by only one line segment in the text. So removing the consecutive decision points improves the reliability of the segmentation.

In order to get the best combination of PSP detection rules, we built a simple semi-automatic prototype for evaluating the rules. The prototype performs the PSP detection process on a selected data set word by word. The usage of such prototype allows reviewing the satisfied/Unsatisfied rules for each point in the online text.

Figure 4.13 shows a screenshot of the prototype for enhancing the PSP rules. The image representation of the data sample (word) is shown at the top. The bottom contains the data sample after the preprocessing and the PSP's detection processes. The green square points with red filling show the PSP of each segment of the sample. The unsatisfied rules are listed on the right of the screen. If we click on any point on the polygon, the unsatisfied rules list appears for that point (if it is not selected as a PSP). An example of such a point is highlighted with a black box; the point (468, 121) does not satisfy rules 2, 3 and 5 as is clear from the example.

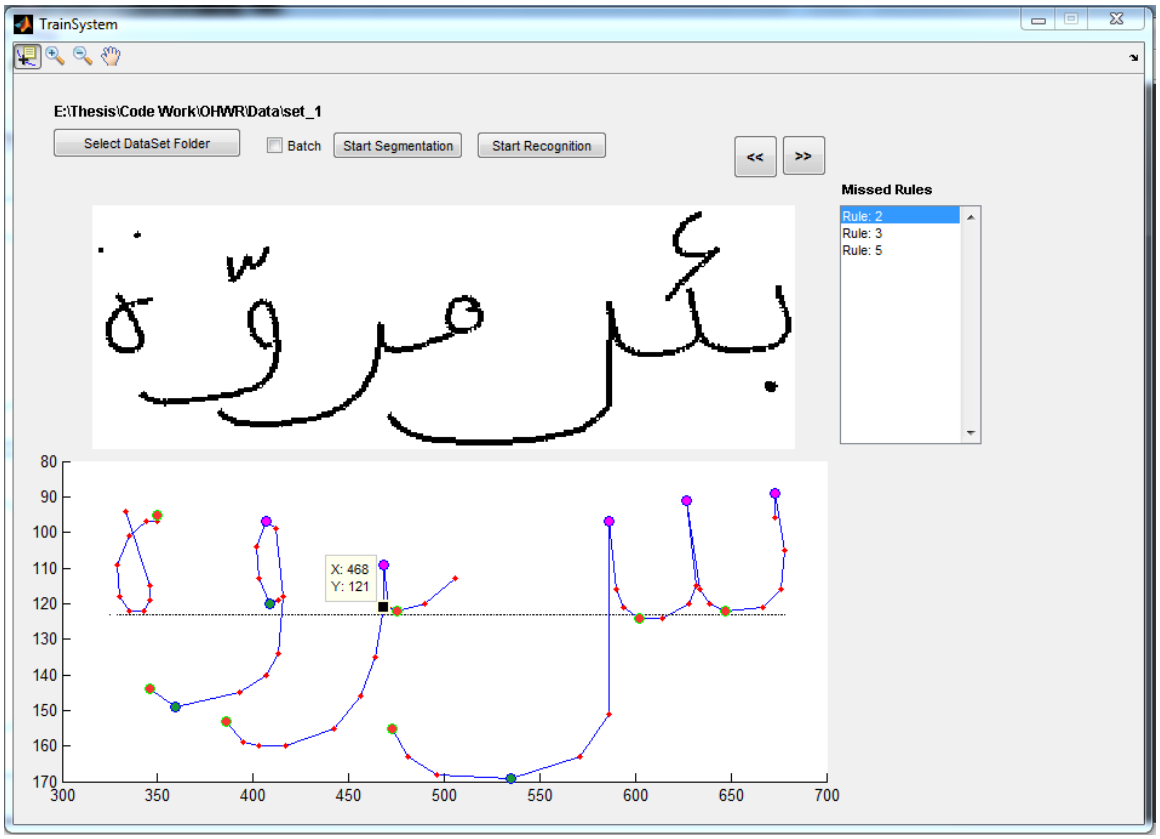


Figure 4.13: Semi-automatic rules' enhancement prototype

This prototype enables us to manually review the evaluation process of the rules. This allows us to update, add or remove the rules based on their behavior. This prototype enables us to perform the PSP detection in batch mode, and export the results to image files so we can perform overall evaluation of the results.

4.4. PSP Detection Problems

The process of PSP detection (Segmentation) is one of the most challenging processes in handwriting recognition. Many factors related to handwriting in general, and to Arabic, in specific lead to this difficulty. One main factor is the many different ways of writing Arabic letters. The shape of the letters changes depending on the way it is written. Figure 4.14 shows an example of four different ways of writing (ـهـ) letter. Such a variety of writing can lead to more complexity in the segmentation process.

Ligatures writing are an example of writing diversity. Some letters like (م) can be written over other letters like (ح). Writing ligatures are one of the most serious challenges in text segmentation for the handwritten and even printed text. Two examples are shown in Figure 4.15 for (محمد، الحج) words. Both words have ligatures.

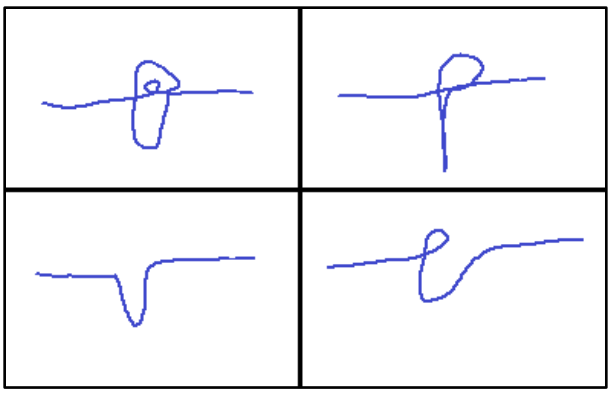


Figure 4.14: Four different ways to write haa' (ھ)

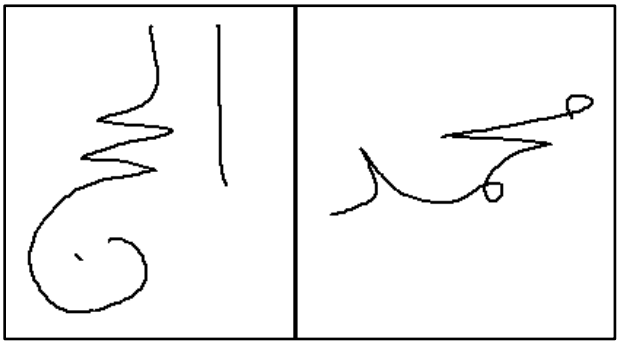


Figure 4.15: Two examples of ligatures

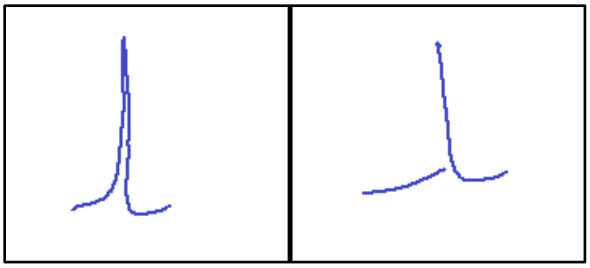


Figure 4.16: Two different shapes of Laam (ـلـ)

Another variety in writing Arabic texts is that each writer may write the same letter in the same writing form but in different ways. The expansion or squeezing of letters and using different number of strokes to write the letters are some examples. Figure 4.16 shows (ﻝ) letter written in two different ways. The letter on the left is written using only one stroke, and two strokes have been used to write the letter on the right.

Different input devices can lead to writing variations that affect the process of PSPs detection process. For example some devices might have low resolution or low sampling rate that results in decreasing the number of points of the text, which may affect recognition.

4.5. Evaluation of PSP Detection

When the PSP detection process takes place, the segments are stored for each data sample (word/sub word). Since we have online datasets, the segments are represented as a list of PSPs. Each two consecutive PSPs bound a set of coordinates that form a letter. Figure 4.1 shows an example of segmented data sample; the PSP marked using a circle around the PSP.

To evaluate the accuracy of the PSP detection process, we used two approaches, semi-automatic and manual evaluation. The main goal of PSP's evaluation process is to count and compare the resulting segments against the ground truth of these segments. Correctly segmented data sample is a sample with number of segments (PSP) equal to the number of letters in the textual representation. In addition, each segment in that sample, should match the real shape/model of the original letter.

In manual evaluation, the tester reviews all the outputs one by one, and then classifies each result as correct or wrong based on his knowledge about the correct decomposition of each data sample (word). The usage of manual evaluation has been used on a limited number of data samples as it requires too much time.

The semi-automatic evaluation approach solves the problem of time, while the accuracy of evaluation decreases. The main challenge of a semi-automatic evaluation approach is that we cannot precisely evaluate its accuracy as we do not have the ground truth segmentation points of the samples.

In addition to the manual approach, a simple evaluation technique has been used in this work. The first step is to evaluate the number of PSPs compared with the number of actual letters in the ground truth table. In general if the number of decision points matches the number of letters in the sample, the segmentation may be classified as correct. This may not be the case from the segmentation point of view, as in some cases extra PSPs are added which can be considered correct although it does not match any letter in the ground truth. On the other hand some PSP might be correctly removed from the segmentation logic while it should not be removed according to the ground truth text.

An example of such cases is the (ـسـ) seen letter. It is one letter and hence it should be one PSP to describe it. But from the segmentation point of view this letter may be segmented into 3 segments and 3 PSPs will be produced. Figure 4.17 Show the word (الرسالة) segmented correctly from the segmentation point of view, while it has 2 extra PSPs compared to the labels count (decision point are pointed).

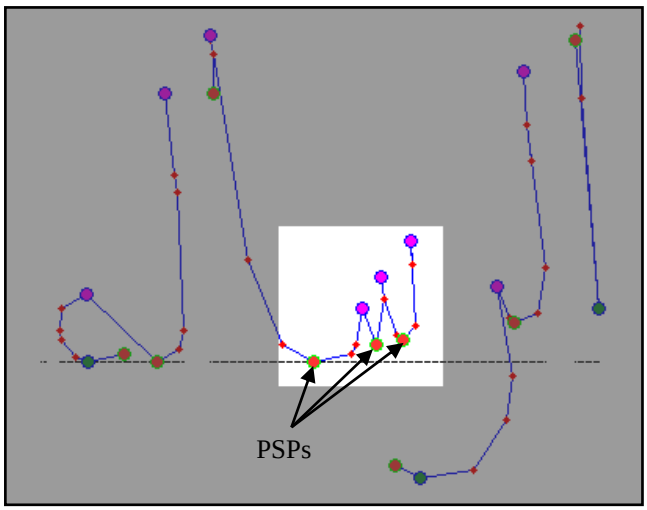


Figure 4.17: The seen (سـ) letter gives extra PSPs

In this example we can see that our segmentation algorithm worked correctly by adding extra PSP, as the algorithm cannot differentiate between a seen (ـسـ) letter and three baa (ـبـ) taa (ـتـ) or thaa (ـثـ) letters. This means that the occurrence of seen (ـسـ) letter can have a similar shape of the other two or more letters' occurrence. To overcome this issue a positive and negative threshold is used. The usage of the thresholds enables the PSP detection to add or remove PSPs based on the letters existing in the ground truth values. For a data sample L which has *n* letters, the final number of PSPs of that label may be:

$$n + postiveThreshold \geq number\ of\ DP \geq n - minusThreshold$$

The positive and negative threshold for each letter is:

$$postiveThreshold, minusThreshold = nc * av$$

Where *nc* is the number of occurrences of the letter and *av* is the threshold of that letter. Table 5 contains the letters and their thresholds.

Table 5: Allowance letters with their values

Letter_En	Letter_Ar	Threshold+/-
Seen	س	+2
Sheen	ش	+2
Saad	ص	+1
Daad	ض	+1
TTAA	ط	+1
TTHAA	ظ	+1
Alef after Laam	لا	-1

If the number of PSP's is satisfied, the segmentation is considered correct; otherwise it is considered wrong.

4.6. Experimental Results

The PSP's detection process has been tested using 4 datasets. One dataset is collected for this project and the other datasets represents the ADAB database of online handwriting recognition. The first dataset (Our Data) contains 193 samples of texts composed from one to three words. The ADAB dataset is divided into three parts (set1, set2 and set3), each set contains samples of one to 4 words. We have done the manual evaluation on a partial set of the ADAB database as its size is very large and hence very time consuming.

Table 6 shows the size of each dataset and the segmentation accuracy. It also shows the number of correctly/miss-segmented samples in each set.

Table 6: PSP detection results of different datasets

Dataset	Dataset Size	Evaluated Samples	Accuracy	
			Manual %	Semi-Automatic %
Our Set	193	193	84.91%	53.6%
ADAB Set1	5037	5037	87.8%	40.10%
ADAB Set2	5090	5090	81.4%	42.33%
ADAB Set3	5037	519	82.9%	43.15%

CHAPTER 5

CHARACTERS' FUZZY MODELING & RECOGNITION

In the previous chapter, we discussed the PSP detection of Arabic online text. We presented our fuzzy based technique for modeling Arabic letters. This technique uses the directions and lengths of the polygon segments to model Arabic letters. To do so, two new approaches have been introduced in this thesis. Model Trend Line (MTL) and Model Envelop (ME).

5.1. Fuzzy Attributed Turning Function

The output of PSP detection of Arabic online text is a set of segmented letters (segments). Each letter is represented by a polygon which is composed of sequence of (x, y) coordinates that represent the dominant points that form the shape of the letter. Let $D_i = (x_i, y_i)$, i=1, 2, …,n_d represent that sequence, these points represent a polygon A of n_d edges, each edge d_i is a vector with length $l_i=|d_i|$ where $d_i=d_{i+nd}$. These polygons are represented by turning function [32]. For polygon A, the turning function $\theta_A(s)$ gives the angle of the counterclockwise tangents at each point of that polygon. It is called cumulative angel function. The value of the x-axis of that function represents the length of the segment. Figure 5.1 illustrates an example of the turning function for the Arabic letter Waaw (و).

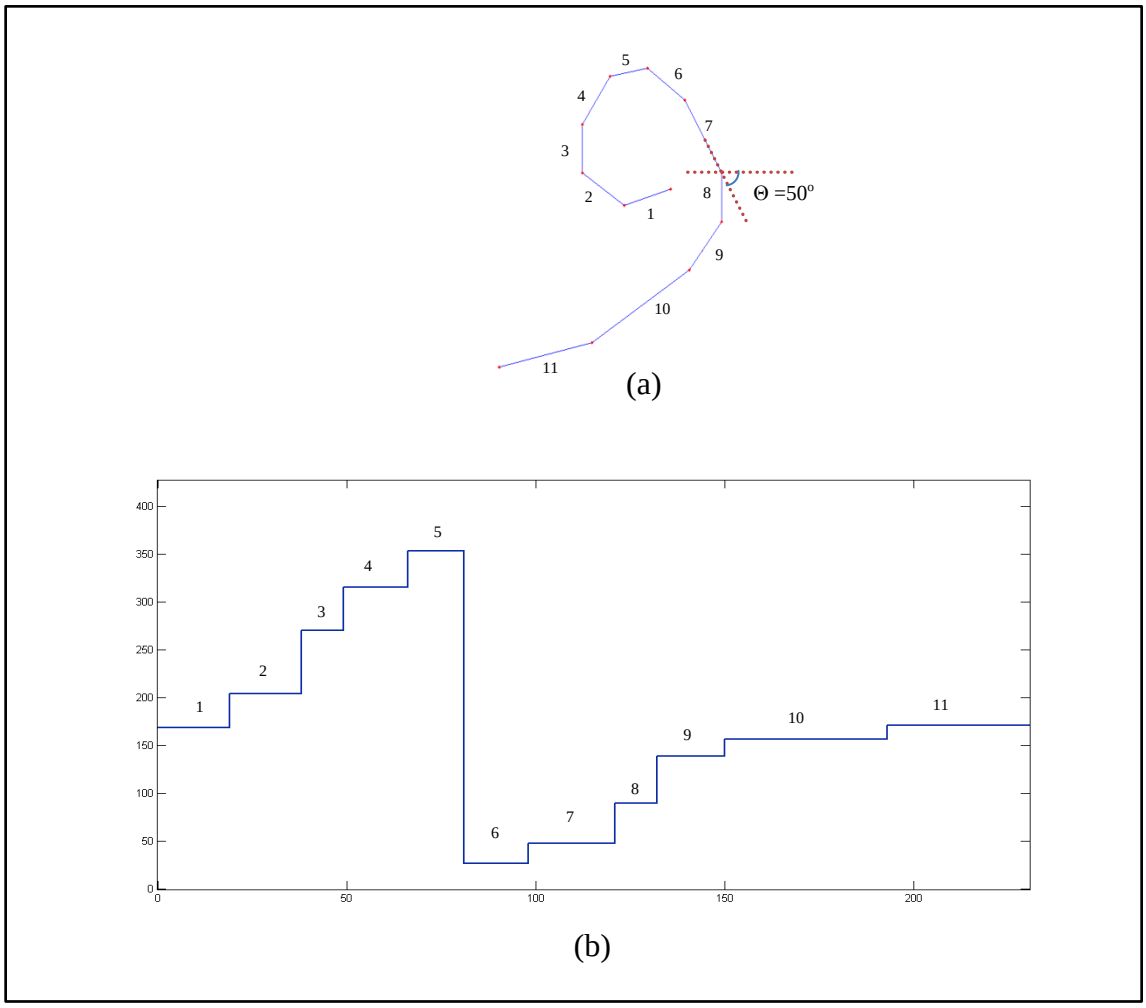


Figure 5.1: Illustration of turning function (a) The polygonal representation of Arabic letter Waaw (و). (b) The turning function of (a)

5.2. Fuzzy Models

The turning function representation of letters reflects its shape. This representation is needed to build generic models that represent sets of similar models. Column (**C**) of Table 3 shows the 54 different models that represent all shapes of Arabic letters (108 shapes). Examples of those models are shown in section 3.2. Building generic models that combine letters of similar shape reduces the number of models that are needed to represent all Arabic letters. These models are represented by fuzzy models with directions and lengths of the line segments of the characters. Two types of models have been built to represent the models of the letters. MTL and ME are built for each class. The models are generated from the aggregated samples of the letters in the training phase.

5.2.1. Letters' Samples

In order to generate the models of each class, a set of samples of Arabic letters have been collected. Figure 5.2 (a) shows samples of letters in the database. Each form of the letter has 8 samples written by 8 different writers. Figure 5.2 (b) shows samples of Arabic letter TTAA (ط). The number of samples in building each model exceeds the number of writers. This is a result of using samples of similar letters for the same class. For instance, Arabic letters (ـخ ،ـح ،ـج) have the same primary shape. So we use the samples of the three letters to build the model of that shape. Hence, twenty four samples are used to build the models of these letters.

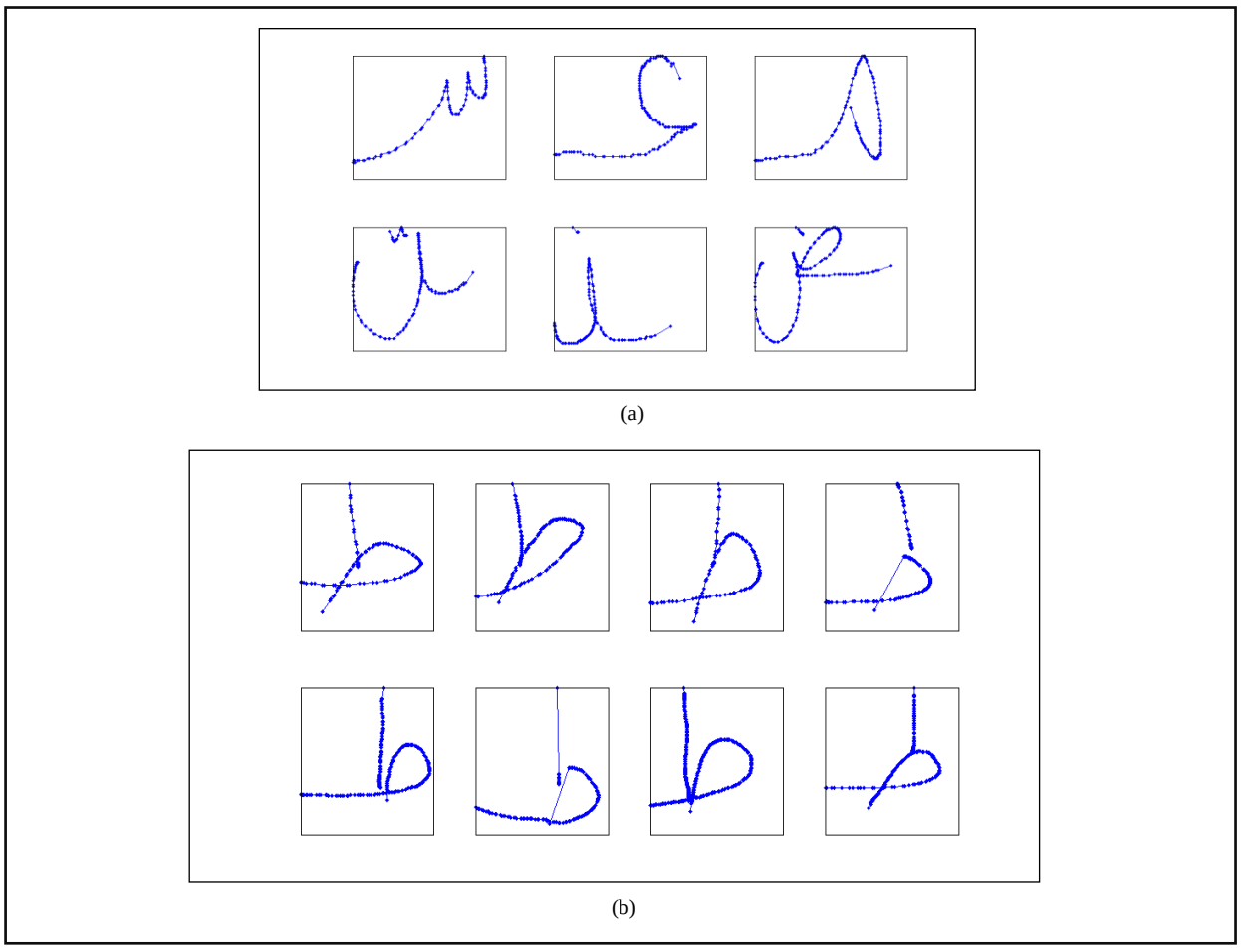


Figure 5.2: Examples form the Letters database. (a) Samples of some letters (b) Eight Samples of Arabic letter TTAA (ط)

The polygonal approximation of each sample has been generated. Then the directions and lengths of each polygon are calculated. These directions are used to generate the fuzzy models for the letters' classes. The lengths of all letters' samples have been normalized to a unique length. The normalization aligns the samples to the same length. Figure 5.3 shows two samples normalized to a length of 700. We can see that both of the models have similar structure. In this work we normalized all the samples' lengths to 700. This length is derived based on the common lengths of samples. We found that this length is enough to model the samples.

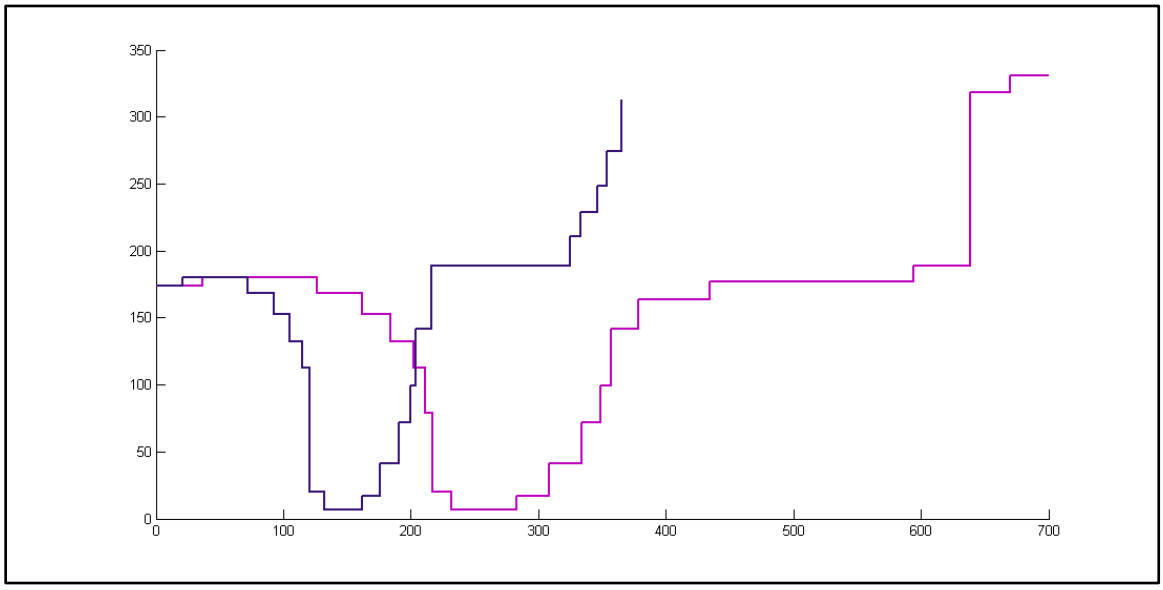


Figure 5.3: Two similar samples one of them normalized to the length of 700

After normalizing the samples, turning function model of the polygonal approximation is created for all samples of each class. Figure 5.4 shows samples of the Arabic letter Baa (ـبـ) class. The figure shows 40 samples for the letter written by different writers. We can notice that all the samples have a trend line. The extraction of this trend line is done in two approaches in order to get the best model. In the first approach, MTL, a single curve that represents the mean of all samples is presented. The other approach, ME, where the top and bottom envelops for the aggregated samples is used. Top and bottom envelops of the samples form a good model to describe the shape of the letters.

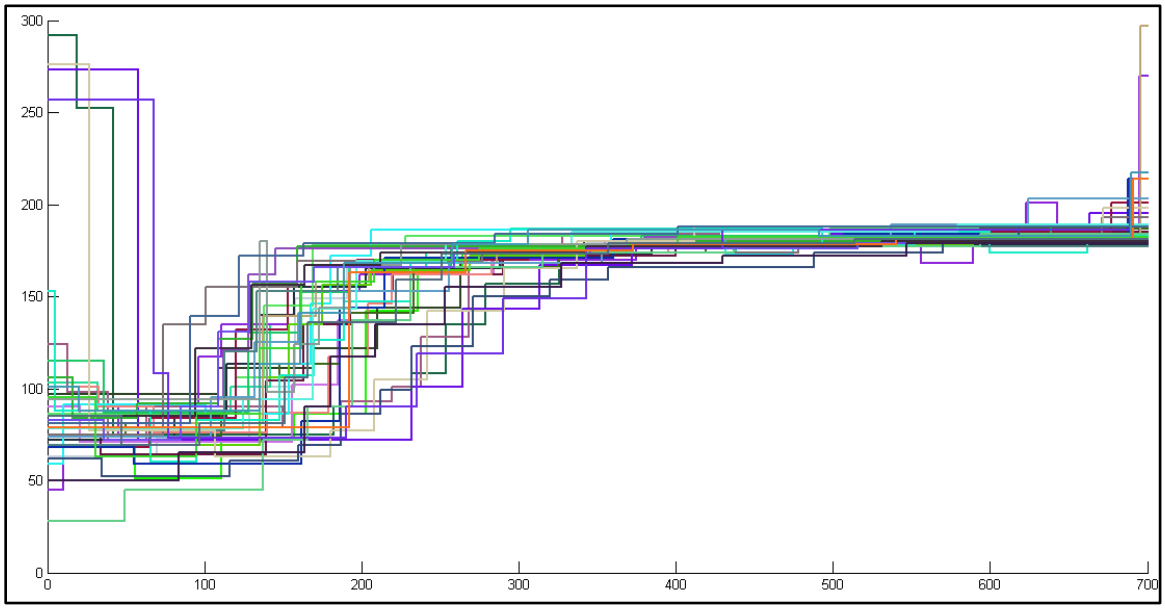


Figure 5.4: Aggregation of samples of the letter Baa (ﺑ) Class

5.2.2. Model Trend Line

MTL is a curve that represents the center of the aggregated samples and hence the general shape of the class. The aggregation is done by using a moving window over the samples and estimating the mean at each window location (sampling point):

$$TL_m(s) = \frac{1}{n_s} \sum_{i=1}^{n_s} y_i$$

Where *s* is the line segment index, n_s number of points in segment *s*. y_i is the *y*-coordinate of point *i* of the segment. The standard deviation is given by the following equation:

$$TL_s = \sqrt[2]{\frac{1}{n_s} \sum_{i=1}^{n_s} (Y_i - TL_m(s))^2}$$

The trend line has t=1…l/w sampling points, l is taken as 700 for normalization. The window size w is taken based on modest analysis. A value of w=15 is chosen so the final size of the trend line reflects the actual trend of samples.

The MTL is series of TL (1...t) with the Mean and the Standard Deviation of *y*-values of all samples at each window. Figure 5.5 shows an example of the MTL for the Arabic letter Baa (ـبـ), the window *w* is shown on the figure.

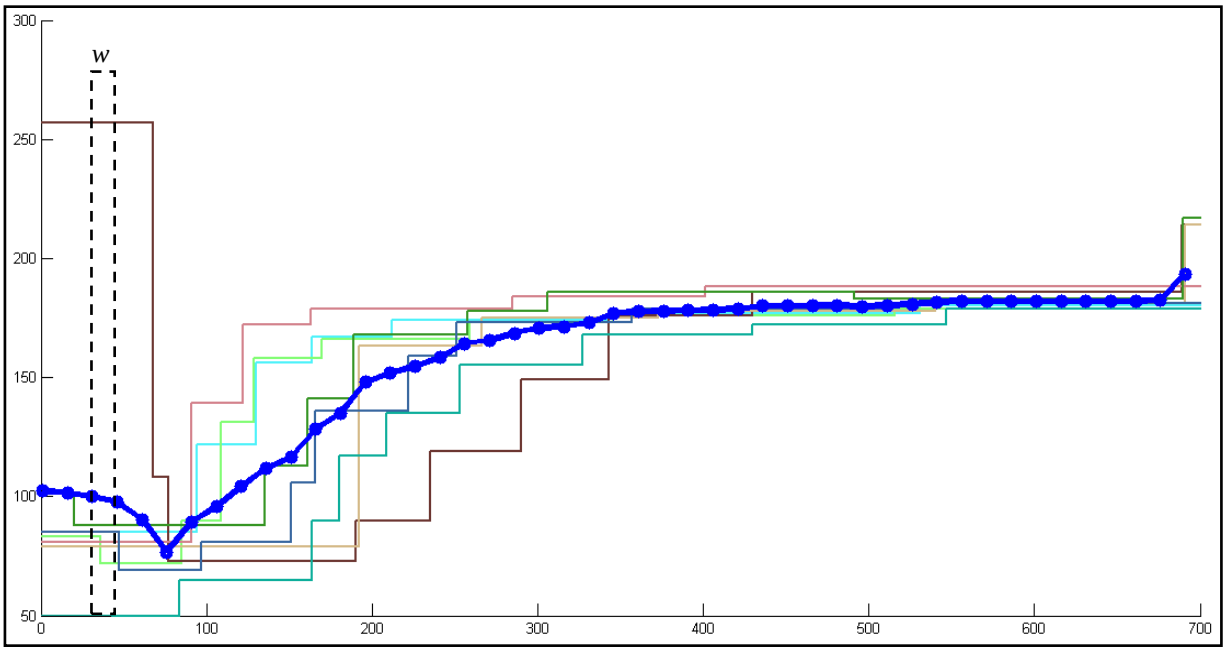


Figure 5.5: Model Trend Line (MTL) of the Arabic letter Baa (ب)

This representation shows the general shape of the letter. It represents both the angels and lengths of the letter segments.

5.2.3. Model Envelop (ME)

ME is represented by two curves that surround the main body of the samples aggregation. The top and bottom envelops describe the range and shape of each letter main body.

Every sample is represented as a function y = C(x). Given a set C = {C1, C2, …, Cn } of x- curves that represent n samples. The bottom envelope is defined as the point-wise minimum of all samples. The bottom envelope for the set C at position *i* can be defined as follows:

$$B_c[i] = minC_i(x)$$

$$where\ x = 1, 2, \ldots nubmer\ of\ samples, i = 1, 2, \ldots 700$$

Similarly, the top envelope of C is the point-wise maximum of the samples curves in the set:

$$T_c[i] = maxC_i(x)$$

$$where\ x = 1, 2, \ldots nubmer\ of\ samples, i = 1, 2, \ldots 700$$

Figure 5.7 (a) demonstrates the aggregated samples that form the Arabic Letter Faa and Qaf (ف، ق). The top envelop, shown in black, shows the upper angel limit of the model. The bottom envelop, shown in red dashed line, shows the lower limit of the angels. Figure 5.7 (b) shows the ME of the Arabic letter Faa (ف) and Qaf (ق) class. In order to enhance the reliability of the envelops, simple rlowess (Robust Locally Weighted

Scatterplot Smoothing) smoothing [33] has been applied. The smoothing removes the noisy variations of envelops. Figure 5.6 shows a part of the ME in solid line before smoothing and the dashed line after smoothing.

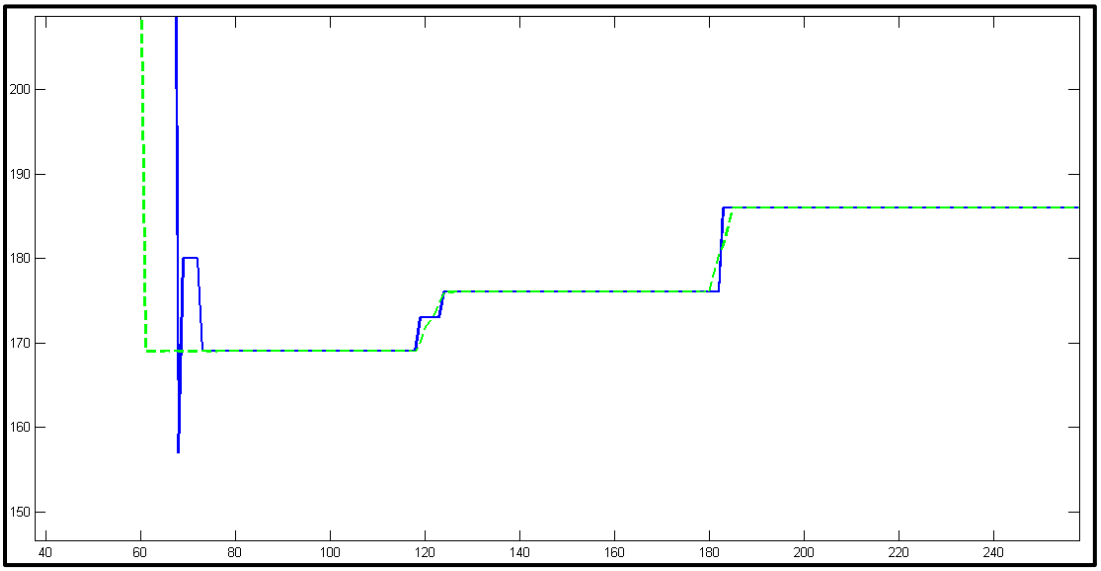


Figure 5.6: Part of the model envelop before and after smoothing

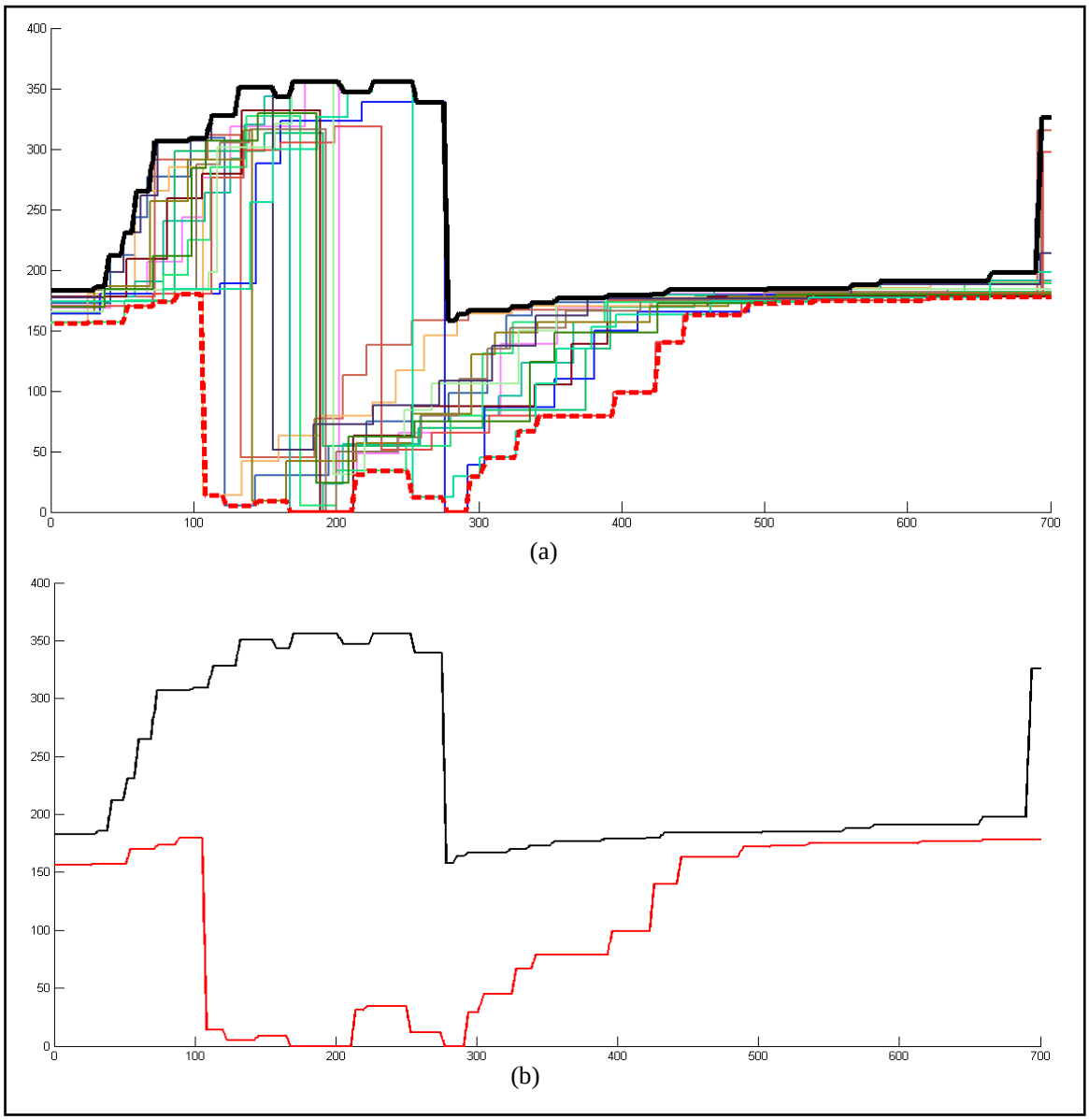


Figure 5.7: Models' samples with top and bottom envelop of Arabic letter Faa (ڤ). (b) The final model envelop for (a)

In the next section we present the properties and analyses of the MTL and ME.

5.3. Models Analysis and Discussion

MTL and ME are used differently. In order to estimate the dissimilarity of the models, similarity matrices for the MTL and the ME are built. These matrices show the overlap of the models of the letters' classes. In addition, it shows which models are confused with other models.

5.3.1. MTL vs. Model Envelops Similarity

The similarity matrix between the MTL and ME of all letters' models has been built. This matrix shows the uniqueness of each model. The size of matrix is 54×54 based on the total number of models. Each coordinate of the matrix shows the similarity between the MTL and the ME. Similarity values above 95% are marked with red color. We compare the ME vs. MTL. Similar models are grouped and analyzed. Figure 5.8 shows the similarity matrix between the MTL and ME. This matrix indicates the suitability of the ME to model the classes of the letters. We have noted a number of similar models. Mainly the model numbered 13 (letters: ﺧ ،ﺣ ،ﺟ) and 29 (letters: ﺍ) showed the highest similarity with large number of other models. Table 7 is reporting the models with high similarity with other models. This shows large number of similar models, which may indicate that the ME is not a suitable model to represent the letters. Note that the diagonal is the similarity between the models and itself. Table 8 shows the 54 models and their shapes.

Table 7: Similar models based on ME -MTL comparison

Model Number	Similar To (Model Number)	Similarity Values (0..1)
2	1	1
10	3, 4, 5	1, 1, 0.96
13	1, 2, 3, 5, 7, 8, 10, 12	0.98, 1, 0.96, 1, 0.96, 0.96, 0.96, 1
16	12	1
20	12, 16	1, 0.98
22	14	0.96
24	12, 16, 20	0.97, 0.98, 1
25	19	0.96
27	24	0.98
29	Most of other Models	>0.96
34	12, 16, 17, 20, 22	0.97, 0.96, 0.96, 0.98, 0.95
38	27	0.96
42	7	0.98
44	40	0.98
45-51	18, 46	0.96..1
52	11	0.97
53	48	0.96
54	46, 50, 51	0.96, 0.98, 0.96

Table 8: Models' Numbers and Shapes

Model No	Model Shape	Model No	Model Shape	Model No	Model Shape
1	د	19	ع	37	ع
2	ح	20	ف	38	ف
3	س	21	Empty	39	ل
4	ص	22	ل	40	م
5	ع	23	م	41	ن
6	و	24	ں	42	ه
7	ک	25	ه	43	و
8	ل	26	و	44	لا
9	م	27	ی	45	ء
10	ھ	28	لا	46	ـہ
11	ا	29	ا	47	ط
12	ب	30	ں	48	س
13	ح	31	ح	49	ص
14	د	32	د	50	ھ

15	ـر	33	ر	51	ـه
16	ـس	34	س	52	ـك
17	ـص	35	ص	53	ـم
18	ـط	36	ط	54	ـھ

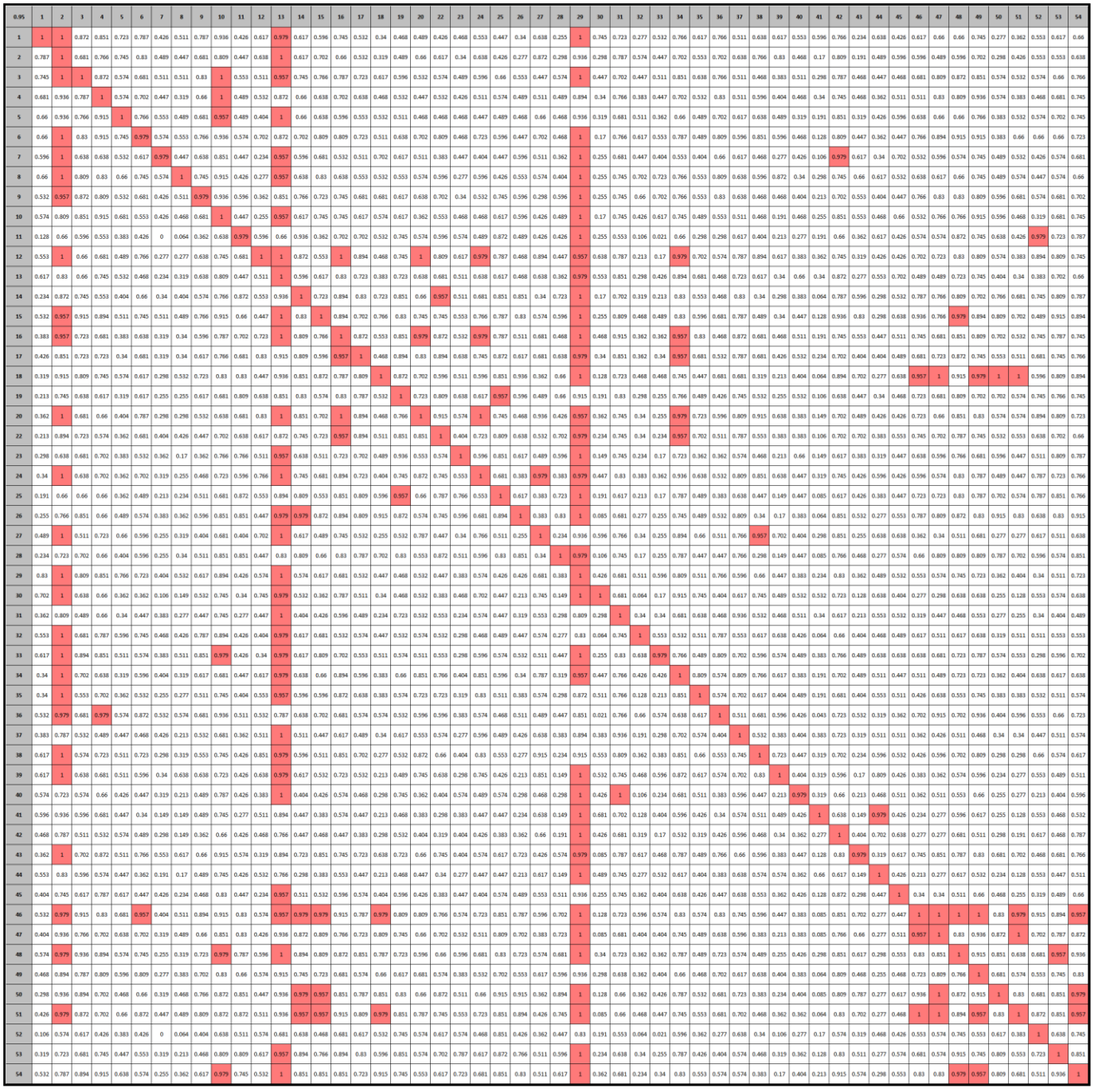


Figure 5.8: MTL vs. Model Envelop confusion matrix

5.3.2. Classes MTL vs. MTL

In order to evaluate the adequacy of the MTL models we have created a MTL vs. MTL confusion matrix in order to show the number of similar models. Figure 5.9 shows the confusion matrix. The similarity between models is shaded in red. Table 9 is listing the most similar models and the similarity ratio between those models. Note that the diagonal shows the similarity between the models of the same class.

0.95	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
1	1.000	0.903	0.943	0.849	0.892	0.871	0.843	0.916	0.908	0.888	0.797	0.879	0.900	0.848	0.892	0.864	0.843	0.856	0.818	0.853	0.809	0.817	0.847	0.819	0.851	0.844	0.813	0.934	0.920	0.816	0.900	0.950	0.868	0.822	0.802	0.820	0.852	0.912	0.837	0.899	0.833	0.827	0.870	0.872	0.895	0.848	0.915	0.868	0.853	0.867	0.742	0.872	0.902
2	0.903	1.000	0.875	0.873	0.945	0.901	0.870	0.920	0.834	0.904	0.769	0.860	0.887	0.835	0.892	0.858	0.830	0.832	0.799	0.868	0.820	0.808	0.874	0.790	0.840	0.907	0.794	0.926	0.837	0.840	0.907	0.915	0.850	0.835	0.844	0.815	0.890	0.917	0.838	0.830	0.850	0.858	0.830	0.917	0.881	0.828	0.882	0.859	0.839	0.851	0.714	0.866	0.870
3	0.943	0.875	1.000	0.886	0.872	0.904	0.858	0.881	0.925	0.891	0.824	0.906	0.887	0.887	0.922	0.898	0.880	0.890	0.859	0.885	0.852	0.849	0.869	0.859	0.886	0.828	0.863	0.909	0.902	0.783	0.861	0.922	0.894	0.826	0.844	0.796	0.854	0.866	0.833	0.849	0.821	0.872	0.822	0.872	0.927	0.885	0.953	0.897	0.895	0.902	0.770	0.903	0.932
4	0.849	0.873	0.886	1.000	0.862	0.907	0.817	0.818	0.830	0.904	0.776	0.858	0.843	0.824	0.872	0.841	0.830	0.827	0.808	0.839	0.814	0.817	0.836	0.808	0.829	0.812	0.810	0.855	0.805	0.767	0.790	0.843	0.818	0.855	0.918	0.740	0.870	0.799	0.828	0.766	0.792	0.872	0.753	0.859	0.866	0.832	0.897	0.857	0.839	0.841	0.723	0.868	0.878
5	0.892	0.945	0.872	0.862	1.000	0.891	0.869	0.907	0.816	0.909	0.776	0.847	0.871	0.812	0.874	0.832	0.817	0.827	0.787	0.845	0.802	0.829	0.853	0.800	0.823	0.905	0.780	0.926	0.826	0.794	0.897	0.891	0.825	0.796	0.822	0.781	0.873	0.882	0.812	0.817	0.852	0.857	0.810	0.942	0.864	0.823	0.884	0.842	0.822	0.836	0.720	0.880	0.885
6	0.871	0.901	0.904	0.907	0.891	1.000	0.880	0.872	0.877	0.883	0.797	0.887	0.849	0.863	0.910	0.880	0.869	0.849	0.835	0.901	0.887	0.848	0.900	0.821	0.867	0.867	0.846	0.910	0.823	0.758	0.844	0.876	0.872	0.813	0.919	0.769	0.884	0.850	0.805	0.780	0.843	0.946	0.774	0.878	0.906	0.853	0.912	0.885	0.872	0.875	0.742	0.894	0.888
7	0.843	0.870	0.858	0.817	0.869	0.880	1.000	0.890	0.821	0.840	0.730	0.823	0.832	0.854	0.909	0.848	0.810	0.833	0.791	0.855	0.853	0.802	0.854	0.775	0.875	0.846	0.820	0.892	0.793	0.753	0.832	0.867	0.846	0.792	0.818	0.777	0.817	0.827	0.783	0.750	0.830	0.899	0.743	0.873	0.871	0.830	0.859	0.833	0.873	0.870	0.675	0.851	0.848
8	0.916	0.920	0.881	0.818	0.907	0.872	0.890	1.000	0.865	0.862	0.723	0.815	0.852	0.822	0.890	0.833	0.797	0.819	0.779	0.847	0.821	0.788	0.847	0.763	0.836	0.872	0.784	0.943	0.842	0.817	0.933	0.945	0.856	0.787	0.808	0.808	0.839	0.933	0.792	0.841	0.833	0.877	0.842	0.892	0.863	0.807	0.852	0.827	0.834	0.841	0.667	0.838	0.839
9	0.908	0.834	0.925	0.830	0.816	0.877	0.821	0.865	1.000	0.826	0.801	0.867	0.850	0.887	0.888	0.869	0.871	0.878	0.859	0.857	0.864	0.810	0.824	0.845	0.884	0.778	0.873	0.867	0.869	0.766	0.851	0.900	0.860	0.786	0.835	0.792	0.795	0.843	0.789	0.815	0.768	0.858	0.791	0.818	0.920	0.872	0.901	0.884	0.895	0.891	0.745	0.858	0.876
10	0.888	0.904	0.891	0.904	0.909	0.883	0.840	0.862	0.826	1.000	0.805	0.850	0.895	0.846	0.893	0.857	0.843	0.865	0.824	0.833	0.814	0.845	0.836	0.839	0.854	0.846	0.823	0.890	0.825	0.805	0.834	0.891	0.825	0.847	0.834	0.794	0.880	0.837	0.862	0.809	0.832	0.857	0.802	0.914	0.878	0.858	0.905	0.864	0.854	0.861	0.750	0.895	0.914
11	0.797	0.769	0.824	0.776	0.776	0.797	0.730	0.723	0.801	0.805	1.000	0.872	0.809	0.842	0.809	0.842	0.836	0.857	0.822	0.838	0.799	0.787	0.837	0.874	0.813	0.789	0.803	0.769	0.838	0.693	0.702	0.753	0.817	0.779	0.740	0.735	0.807	0.768	0.717	0.813	0.839	0.751	0.811	0.768	0.857	0.859	0.863	0.845	0.832	0.850	0.903	0.863	0.869
12	0.879	0.860	0.906	0.858	0.847	0.887	0.823	0.815	0.867	0.850	0.872	1.000	0.910	0.887	0.901	0.950	0.943	0.864	0.877	0.962	0.898	0.831	0.932	0.885	0.871	0.897	0.839	0.858	0.921	0.815	0.795	0.845	0.933	0.880	0.829	0.847	0.922	0.866	0.798	0.856	0.826	0.844	0.826	0.833	0.913	0.864	0.930	0.888	0.878	0.884	0.847	0.901	0.902
13	0.900	0.887	0.887	0.843	0.871	0.849	0.832	0.852	0.850	0.895	0.809	0.910	1.000	0.872	0.899	0.922	0.903	0.862	0.877	0.895	0.865	0.827	0.879	0.886	0.868	0.881	0.836	0.879	0.897	0.880	0.827	0.889	0.890	0.882	0.794	0.888	0.887	0.871	0.868	0.852	0.812	0.829	0.827	0.873	0.886	0.851	0.900	0.862	0.862	0.866	0.775	0.884	0.895
14	0.848	0.835	0.887	0.824	0.812	0.863	0.854	0.822	0.887	0.846	0.842	0.887	0.872	1.000	0.926	0.913	0.901	0.915	0.907	0.885	0.910	0.823	0.856	0.894	0.964	0.802	0.941	0.832	0.830	0.791	0.784	0.843	0.869	0.848	0.811	0.848	0.820	0.803	0.818	0.766	0.777	0.860	0.743	0.822	0.943	0.919	0.914	0.908	0.958	0.942	0.787	0.883	0.893
15	0.892	0.892	0.922	0.872	0.874	0.910	0.909	0.890	0.888	0.893	0.809	0.901	0.899	0.926	1.000	0.927	0.894	0.915	0.881	0.916	0.907	0.867	0.898	0.863	0.941	0.850	0.891	0.899	0.855	0.812	0.842	0.909	0.909	0.855	0.856	0.843	0.861	0.850	0.848	0.801	0.832	0.918	0.783	0.883	0.946	0.901	0.941	0.913	0.938	0.935	0.753	0.920	0.927
16	0.864	0.858	0.898	0.841	0.832	0.880	0.848	0.833	0.869	0.857	0.842	0.950	0.922	0.913	0.927	1.000	0.932	0.884	0.889	0.950	0.919	0.819	0.931	0.882	0.899	0.880	0.865	0.849	0.901	0.839	0.801	0.859	0.944	0.904	0.819	0.874	0.903	0.862	0.819	0.833	0.809	0.869	0.809	0.835	0.919	0.879	0.912	0.885	0.901	0.903	0.813	0.887	0.894
17	0.843	0.830	0.880	0.830	0.817	0.869	0.810	0.797	0.871	0.843	0.836	0.943	0.903	0.901	0.894	0.932	1.000	0.858	0.896	0.925	0.927	0.838	0.888	0.895	0.884	0.847	0.879	0.828	0.871	0.818	0.775	0.832	0.898	0.846	0.820	0.858	0.873	0.827	0.806	0.806	0.783	0.842	0.782	0.817	0.894	0.850	0.902	0.868	0.878	0.867	0.806	0.893	0.881
18	0.856	0.832	0.890	0.827	0.827	0.849	0.833	0.819	0.878	0.865	0.857	0.864	0.862	0.915	0.915	0.884	0.858	1.000	0.887	0.856	0.850	0.828	0.831	0.905	0.915	0.782	0.907	0.838	0.820	0.758	0.780	0.851	0.843	0.816	0.797	0.797	0.804	0.783	0.815	0.756	0.765	0.846	0.730	0.827	0.939	0.975	0.916	0.918	0.930	0.948	0.782	0.878	0.915
19	0.818	0.799	0.859	0.808	0.787	0.835	0.791	0.779	0.859	0.824	0.822	0.877	0.877	0.907	0.881	0.889	0.896	0.887	1.000	0.864	0.869	0.864	0.830	0.935	0.899	0.781	0.899	0.801	0.813	0.810	0.749	0.814	0.843	0.805	0.785	0.836	0.808	0.772	0.850	0.743	0.740	0.821	0.724	0.802	0.906	0.880	0.891	0.890	0.901	0.893	0.779	0.863	0.870
20	0.853	0.868	0.885	0.839	0.845	0.901	0.855	0.847	0.857	0.833	0.838	0.962	0.895	0.885	0.916	0.950	0.925	0.856	0.864	1.000	0.925	0.823	0.950	0.861	0.878	0.906	0.835	0.870	0.892	0.810	0.808	0.850	0.952	0.875	0.847	0.851	0.907	0.874	0.781	0.828	0.832	0.876	0.807	0.834	0.914	0.854	0.902	0.882	0.878	0.885	0.815	0.898	0.877
22	0.809	0.820	0.852	0.814	0.802	0.887	0.853	0.821	0.864	0.814	0.799	0.898	0.865	0.910	0.907	0.919	0.927	0.850	0.869	0.925	1.000	0.809	0.880	0.857	0.899	0.835	0.887	0.826	0.823	0.787	0.769	0.818	0.900	0.848	0.844	0.839	0.843	0.810	0.774	0.758	0.778	0.893	0.741	0.803	0.887	0.853	0.869	0.864	0.902	0.882	0.764	0.873	0.846
23	0.817	0.808	0.849	0.817	0.829	0.848	0.802	0.788	0.810	0.845	0.787	0.831	0.827	0.823	0.867	0.819	0.838	0.828	0.864	0.823	0.809	1.000	0.807	0.864	0.839	0.767	0.823	0.833	0.776	0.749	0.758	0.820	0.799	0.738	0.797	0.763	0.781	0.757	0.887	0.726	0.772	0.822	0.705	0.848	0.851	0.819	0.875	0.827	0.831	0.823	0.717	0.902	0.889
24	0.847	0.874	0.869	0.836	0.853	0.900	0.854	0.847	0.824	0.836	0.837	0.932	0.879	0.856	0.898	0.931	0.888	0.831	0.830	0.950	0.880	0.807	1.000	0.831	0.846	0.930	0.803	0.873	0.904	0.806	0.817	0.851	0.938	0.864	0.828	0.832	0.922	0.889	0.774	0.862	0.855	0.869	0.842	0.842	0.887	0.828	0.883	0.858	0.848	0.857	0.818	0.871	0.862
25	0.819	0.790	0.859	0.808	0.800	0.821	0.775	0.763	0.845	0.839	0.874	0.885	0.886	0.894	0.863	0.882	0.895	0.905	0.935	0.861	0.857	0.864	0.831	1.000	0.879	0.785	0.887	0.796	0.828	0.786	0.739	0.801	0.838	0.803	0.770	0.818	0.812	0.775	0.816	0.761	0.745	0.800	0.734	0.803	0.892	0.896	0.894	0.885	0.890	0.894	0.820	0.884	0.895
26	0.851	0.840	0.886	0.829	0.823	0.867	0.875	0.836	0.884	0.854	0.813	0.871	0.868	0.964	0.941	0.899	0.884	0.915	0.899	0.878	0.899	0.839	0.846	0.879	1.000	0.797	0.938	0.848	0.816	0.788	0.789	0.856	0.860	0.832	0.819	0.839	0.808	0.794	0.832	0.753	0.781	0.873	0.732	0.834	0.946	0.914	0.916	0.915	0.964	0.942	0.757	0.890	0.901
27	0.844	0.907	0.828	0.812	0.905	0.867	0.846	0.872	0.778	0.846	0.789	0.897	0.881	0.802	0.850	0.880	0.847	0.782	0.781	0.906	0.835	0.767	0.930	0.785	0.797	1.000	0.746	0.891	0.883	0.825	0.848	0.850	0.895	0.851	0.796	0.831	0.935	0.910	0.766	0.860	0.856	0.835	0.849	0.884	0.834	0.777	0.841	0.818	0.793	0.805	0.773	0.833	0.827
28	0.813	0.794	0.863	0.810	0.780	0.846	0.820	0.784	0.873	0.823	0.803	0.839	0.836	0.941	0.891	0.865	0.879	0.907	0.899	0.835	0.887	0.823	0.803	0.887	0.938	0.746	1.000	0.800	0.776	0.759	0.741	0.815	0.817	0.801	0.798	0.812	0.770	0.745	0.815	0.712	0.727	0.844	0.686	0.801	0.911	0.911	0.887	0.911	0.944	0.926	0.749	0.850	0.869
29	0.934	0.926	0.909	0.855	0.926	0.910	0.892	0.943	0.867	0.890	0.769	0.858	0.879	0.832	0.899	0.849	0.828	0.838	0.801	0.870	0.826	0.833	0.873	0.796	0.848	0.891	0.800	1.000	0.869	0.798	0.907	0.942	0.866	0.797	0.840	0.804	0.866	0.909	0.824	0.847	0.874	0.878	0.838	0.918	0.885	0.829	0.896	0.861	0.845	0.853	0.714	0.874	0.880
30	0.920	0.837	0.902	0.805	0.826	0.823	0.793	0.842	0.869	0.825	0.838	0.921	0.897	0.830	0.855	0.901	0.871	0.820	0.813	0.892	0.823	0.776	0.904	0.828	0.816	0.883	0.776	0.869	1.000	0.809	0.830	0.879	0.918	0.852	0.760	0.833	0.885	0.903	0.777	0.929	0.822	0.783	0.891	0.812	0.858	0.811	0.883	0.836	0.817	0.832	0.817	0.845	0.868
31	0.816	0.840	0.783	0.767	0.794	0.758	0.753	0.817	0.766	0.805	0.693	0.815	0.880	0.791	0.812	0.839	0.818	0.758	0.810	0.810	0.787	0.749	0.806	0.786	0.788	0.825	0.759	0.798	0.809	1.000	0.810	0.830	0.814	0.830	0.722	0.910	0.818	0.836	0.831	0.804	0.721	0.752	0.780	0.791	0.784	0.747	0.786	0.757	0.769	0.766	0.675	0.775	0.780
32	0.900	0.907	0.861	0.790	0.897	0.844	0.832	0.933	0.851	0.834	0.702	0.795	0.827	0.784	0.842	0.801	0.775	0.780	0.749	0.808	0.769	0.758	0.817	0.739	0.789	0.848	0.741	0.907	0.830	0.810	1.000	0.924	0.823	0.751	0.769	0.777	0.823	0.922	0.761	0.835	0.796	0.816	0.840	0.876	0.827	0.770	0.824	0.791	0.786	0.796	0.647	0.811	0.811
33	0.950	0.915	0.922	0.843	0.891	0.876	0.867	0.945	0.900	0.891	0.753	0.845	0.889	0.843	0.909	0.859	0.832	0.851	0.814	0.850	0.818	0.820	0.851	0.801	0.856	0.850	0.815	0.942	0.879	0.830	0.924	1.000	0.871	0.807	0.807	0.827	0.850	0.925	0.835	0.870	0.825	0.858	0.862	0.893	0.886	0.837	0.887	0.854	0.858	0.859	0.697	0.862	0.876
34	0.868	0.850	0.894	0.818	0.825	0.872	0.846	0.856	0.860	0.825	0.817	0.933	0.890	0.869	0.909	0.944	0.898	0.843	0.843	0.952	0.900	0.799	0.938	0.838	0.860	0.895	0.817	0.866	0.918	0.814	0.823	0.871	1.000	0.884	0.812	0.855	0.905	0.890	0.777	0.856	0.817	0.855	0.833	0.823	0.890	0.838	0.876	0.857	0.860	0.867	0.797	0.870	0.859
35	0.822	0.835	0.826	0.855	0.796	0.813	0.792	0.787	0.786	0.847	0.779	0.880	0.882	0.848	0.855	0.904	0.846	0.816	0.805	0.875	0.848	0.738	0.864	0.803	0.832	0.851	0.801	0.797	0.852	0.830	0.751	0.807	0.884	1.000	0.828	0.835	0.893	0.818	0.807	0.802	0.755	0.800	0.770	0.786	0.845	0.825	0.840	0.822	0.838	0.844	0.757	0.806	0.827
36	0.802	0.844	0.844	0.918	0.822	0.919	0.818	0.808	0.835	0.834	0.740	0.829	0.794	0.811	0.856	0.819	0.820	0.797	0.785	0.847	0.844	0.797	0.828	0.770	0.819	0.796	0.798	0.840	0.760	0.722	0.769	0.807	0.812	0.828	1.000	0.714	0.820	0.772	0.776	0.709	0.778	0.904	0.697	0.811	0.855	0.803	0.857	0.838	0.822	0.824	0.686	0.843	0.833
37	0.820	0.815	0.796	0.740	0.781	0.769	0.777	0.808	0.792	0.794	0.735	0.847	0.888	0.848	0.843	0.874	0.858	0.797	0.836	0.851	0.839	0.763	0.832	0.818	0.839	0.831	0.812	0.804	0.833	0.910	0.777	0.827	0.855	0.835	0.714	1.000	0.825	0.828	0.815	0.806	0.738	0.767	0.777	0.785	0.825	0.789	0.805	0.794	0.819	0.810	0.729	0.803	0.798
38	0.852	0.890	0.854	0.870	0.873	0.884	0.817	0.839	0.795	0.880	0.807	0.922	0.887	0.820	0.861	0.903	0.873	0.804	0.808	0.907	0.843	0.781	0.922	0.812	0.808	0.935	0.770	0.866	0.885	0.818	0.823	0.850	0.905	0.893	0.820	0.825	1.000	0.892	0.788	0.853	0.827	0.838	0.842	0.859	0.851	0.805	0.866	0.839	0.811	0.825	0.788	0.845	0.851
39	0.912	0.917	0.866	0.799	0.882	0.850	0.827	0.933	0.843	0.837	0.768	0.866	0.871	0.803	0.850	0.862	0.827	0.783	0.772	0.874	0.810	0.757	0.889	0.775	0.794	0.910	0.745	0.909	0.903	0.836	0.922	0.925	0.890	0.818	0.772	0.828	0.892	1.000	0.762	0.900	0.829	0.815	0.906	0.858	0.833	0.773	0.833	0.805	0.791	0.803	0.721	0.820	0.819
40	0.837	0.838	0.833	0.828	0.812	0.805	0.783	0.792	0.789	0.862	0.717	0.798	0.868	0.818	0.848	0.819	0.806	0.815	0.850	0.781	0.774	0.887	0.774	0.816	0.832	0.766	0.815	0.824	0.777	0.831	0.761	0.835	0.777	0.807	0.776	0.815	0.788	0.762	1.000	0.740	0.740	0.787	0.712	0.834	0.835	0.808	0.847	0.819	0.819	0.816	0.662	0.820	0.845
41	0.899	0.830	0.849	0.766	0.817	0.780	0.750	0.841	0.815	0.809	0.813	0.856	0.852	0.766	0.801	0.833	0.806	0.756	0.743	0.828	0.758	0.726	0.862	0.761	0.753	0.860	0.712	0.847	0.929	0.804	0.835	0.870	0.856	0.802	0.709	0.806	0.853	0.900	0.740	1.000	0.829	0.740	0.943	0.795	0.795	0.748	0.825	0.773	0.754	0.768	0.812	0.794	0.812
42	0.833	0.850	0.821	0.792	0.852	0.843	0.830	0.833	0.768	0.832	0.839	0.826	0.812	0.777	0.832	0.809	0.783	0.765	0.740	0.832	0.778	0.772	0.855	0.745	0.781	0.856	0.727	0.874	0.822	0.721	0.796	0.825	0.817	0.755	0.778	0.738	0.827	0.829	0.740	0.829	1.000	0.817	0.859	0.856	0.816	0.759	0.836	0.796	0.774	0.786	0.799	0.855	0.830
43	0.827	0.858	0.872	0.872	0.857	0.946	0.899	0.877	0.858	0.857	0.751	0.844	0.829	0.860	0.918	0.869	0.842	0.846	0.821	0.876	0.893	0.822	0.869	0.800	0.873	0.835	0.844	0.878	0.783	0.752	0.816	0.858	0.855	0.800	0.904	0.767	0.838	0.815	0.787	0.740	0.817	1.000	0.734	0.858	0.881	0.841	0.882	0.853	0.874	0.868	0.697	0.868	0.858
44	0.870	0.830	0.822	0.753	0.810	0.774	0.743	0.842	0.791	0.802	0.811	0.826	0.827	0.743	0.783	0.809	0.782	0.730	0.724	0.807	0.741	0.705	0.842	0.734	0.732	0.849	0.686	0.838	0.891	0.780	0.840	0.862	0.833	0.770	0.697	0.777	0.842	0.906	0.712	0.943	0.859	0.734	1.000	0.794	0.773	0.722	0.796	0.752	0.731	0.744	0.807	0.776	0.782
45	0.872	0.917	0.872	0.859	0.942	0.878	0.873	0.892	0.818	0.914	0.768	0.833	0.873	0.822	0.883	0.835	0.817	0.827	0.802	0.834	0.803	0.848	0.842	0.803	0.834	0.884	0.801	0.918	0.812	0.791	0.876	0.893	0.823	0.786	0.811	0.785	0.859	0.858	0.834	0.795	0.856	0.858	0.794	1.000	0.863	0.817	0.882	0.843	0.837	0.837	0.712	0.885	0.880
46	0.895	0.881	0.927	0.866	0.864	0.906	0.871	0.863	0.920	0.878	0.857	0.913	0.886	0.943	0.946	0.919	0.894	0.939	0.906	0.914	0.887	0.851	0.887	0.892	0.946	0.834	0.911	0.885	0.858	0.784	0.827	0.886	0.890	0.845	0.855	0.825	0.851	0.833	0.835	0.795	0.816	0.881	0.773	0.863	1.000	0.934	0.950	0.954	0.957	0.964	0.801	0.909	0.931
47	0.848	0.828	0.885	0.832	0.823	0.853	0.830	0.807	0.872	0.858	0.859	0.864	0.851	0.919	0.901	0.879	0.850	0.975	0.880	0.854	0.853	0.819	0.828	0.896	0.914	0.777	0.911	0.829	0.811	0.747	0.770	0.837	0.838	0.825	0.803	0.789	0.805	0.773	0.808	0.748	0.759	0.841	0.722	0.817	0.934	1.000	0.912	0.918	0.934	0.950	0.785	0.874	0.904
48	0.915	0.882	0.953	0.897	0.884	0.912	0.859	0.852	0.901	0.905	0.863	0.930	0.900	0.914	0.941	0.912	0.902	0.916	0.891	0.902	0.869	0.875	0.883	0.894	0.916	0.841	0.887	0.896	0.883	0.786	0.824	0.887	0.876	0.840	0.857	0.805	0.866	0.833	0.847	0.825	0.836	0.882	0.796	0.882	0.950	0.912	1.000	0.929	0.924	0.928	0.808	0.930	0.960
49	0.868	0.859	0.897	0.857	0.842	0.885	0.833	0.827	0.884	0.864	0.845	0.888	0.862	0.908	0.913	0.885	0.868	0.918	0.890	0.882	0.864	0.827	0.858	0.885	0.915	0.818	0.911	0.861	0.836	0.757	0.791	0.854	0.857	0.822	0.838	0.794	0.839	0.805	0.819	0.773	0.796	0.853	0.752	0.843	0.954	0.918	0.929	1.000	0.938	0.935	0.792	0.881	0.907
50	0.853	0.839	0.895	0.839	0.822	0.872	0.873	0.834	0.895	0.854	0.832	0.878	0.862	0.958	0.938	0.901	0.878	0.930	0.901	0.878	0.902	0.831	0.848	0.890	0.964	0.793	0.944	0.845	0.817	0.769	0.786	0.858	0.860	0.838	0.822	0.819	0.811	0.791	0.819	0.754	0.774	0.874	0.731	0.837	0.957	0.934	0.924	0.938	1.000	0.963	0.778	0.882	0.903
51	0.867	0.851	0.902	0.841	0.836	0.875	0.870	0.841	0.891	0.861	0.850	0.884	0.866	0.942	0.935	0.903	0.867	0.948	0.893	0.885	0.882	0.823	0.857	0.894	0.942	0.805	0.926	0.853	0.832	0.766	0.796	0.859	0.867	0.844	0.824	0.810	0.825	0.803	0.816	0.768	0.786	0.868	0.744	0.837	0.964	0.950	0.928	0.935	0.963	1.000	0.790	0.883	0.911
52	0.742	0.714	0.770	0.723	0.720	0.742	0.675	0.667	0.745	0.750	0.903	0.847	0.775	0.787	0.753	0.813	0.806	0.782	0.779	0.815	0.764	0.717	0.818	0.820	0.757	0.773	0.749	0.714	0.817	0.675	0.647	0.697	0.797	0.757	0.686	0.729	0.788	0.721	0.662	0.812	0.799	0.697	0.807	0.712	0.801	0.785	0.808	0.792	0.778	0.790	1.000	0.793	0.801
53	0.872	0.866	0.903	0.868	0.880	0.894	0.851	0.838	0.858	0.895	0.863	0.901	0.884	0.883	0.920	0.887	0.893	0.878	0.863	0.898	0.873	0.902	0.871	0.884	0.890	0.833	0.850	0.874	0.845	0.775	0.811	0.862	0.870	0.806	0.843	0.803	0.845	0.820	0.820	0.794	0.855	0.868	0.776	0.885	0.909	0.874	0.930	0.881	0.882	0.883	0.793	1.000	0.941
54	0.902	0.870	0.932	0.878	0.885	0.888	0.848	0.839	0.876	0.914	0.869	0.902	0.895	0.893	0.927	0.894	0.881	0.915	0.870	0.877	0.846	0.889	0.862	0.895	0.901	0.827	0.869	0.880	0.868	0.780	0.811	0.876	0.859	0.827	0.833	0.798	0.851	0.819	0.845	0.812	0.830	0.858	0.782	0.880	0.931	0.904	0.960	0.907	0.903	0.911	0.801	0.941	1.000

Figure 5.9: MTL vs. MTL confusion matrix

Table 9: Similar models based on MTL-MTL

Model Number	Similar To (Model Number)	Similarity Values (0..1)
20	12, 16	0.96, 0.95
26	14	0.96
33	1	0.95
34	20	0.95
47	18	0.97
48	3, 46	0.95, 0.95
49	46	0.95
50	14, 26, 46	0.95, 0.96, 0.95
51	46, 47, 50	0.96, 0.95, 0.96
54	48	0.96

5.3.3. Models' Evaluation

The evaluation of ME and MTL models aims to determine the robustness of each approach. The approach with minimum similarity between models (inter models similarities) is better. An evaluation of both confusion matrices and comparison of similar models is conducted. Using the ME to represent the models, large number of models had high similarity. However, MTL results in limited similarity between models. These models are evaluated manually. It is noted that similarities between these models reflect real similarity in the shape of the letters. Figure 5.10 shows a set of similar models and their corresponding characters.

Model No	Similar to	Model No	Extra Differentiating Features
20		12	
			Loops Number of Dots Position of Body
48		46	
			Number of Notches Number of Dots Position of Dots

Figure 5.10: Examples of similar models and their corresponding letters.

These inter models similarities can be resolved in a post processing phase that follows the recognition phase. Most of the letters in each model have extra features that discriminate them from the other letters. Figure 5.10 shows the extra features that can be used to discriminate between the characters in the post processing phase.

Based on this analysis, the MTL has been chosen to represent the letters' models. This approach is used in the training, and evaluation phases.

CHAPTER 6

ONLINE ARABIC TEXT RECOGNITION

In this chapter we present the proposed prototype for Arabic online text recognition. This chapter introduces the main logic of the recognition phase and describes the developed fuzzy models.

6.1. An Arabic Text Recognition Model

The main logic of the prototype consists of two stages. The characters' modeling stage, in which the system generates different models to describe the shapes of Arabic letters; and the text recognition phase, where the text samples are compared with the models and the most similar models' label are chosen as the label of the text sample. Figure 6.1 shows the sequence of these phases. The left side of the figure demonstrates the characters' modeling phase.

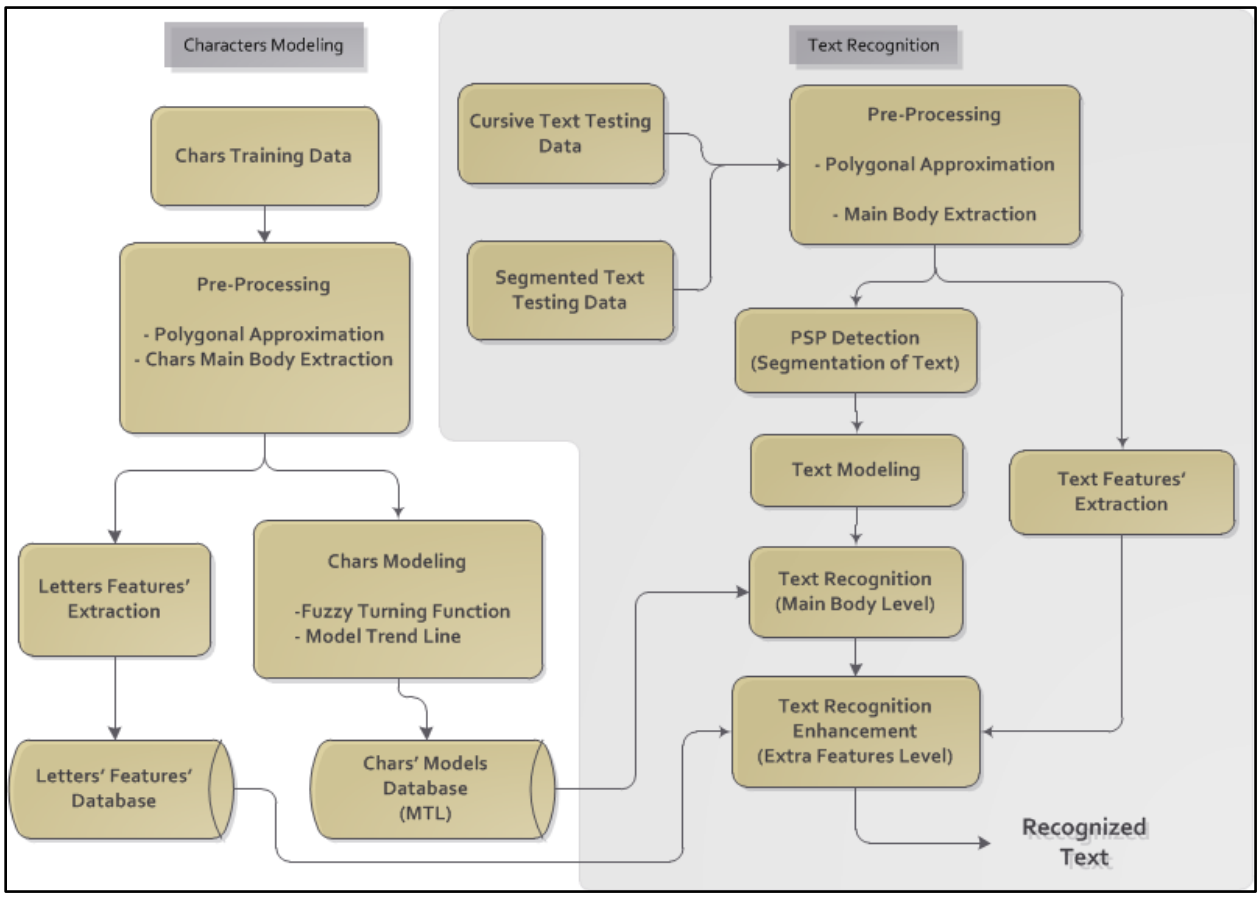


Figure 6.1: Arabic online text recognition model

6.1.1. Characters' Modeling

To model Arabic letters we collected a dataset of isolated characters. This dataset covers all the shapes of Arabic letters. Section 3.1 discuss Arabic writing and the shapes of Arabic text. This data must be captured in online manner using any digital handwriting capturing media supported with stylus. Section 3.2 and 5.2.1 show the collected data and its details. Some preprocessing steps have been applied on the collected data. These preprocessing steps prepare the samples to make them appropriate as an input for the feature extraction phase. Normalization is an initial and basic operation in preprocessing. We scale down or up the letter/word in x-coordinate or y-coordinate. Usually the normalization process is done to make a uniform representation of the data size among all of the samples in the dataset. The second phase of preprocessing is smoothing. Smoothing removes the small curve variations and noise. The third main preprocessing step is the polygonal approximation where we convert the (X, Y) sequence that represents the letters, into polygonal shape that describes this sequence. The algorithm finds the cut points (point of interest) in each curve iteratively. Section 3.3 discussed the polygonal approximation and its importance to the system.

At that point, Letters/Characters main body extraction has been done. The main body of the Arabic letters is separated from the other extra parts of text. Section 4.2 discussed the details of the process of extracting the letters extras and attachments.

After preprocessing, two main types of features' extraction are performed. The first is called Character Modeling is conducted that builds fuzzy models to represent the shape of Arabic letters. Model Trend Line (MTL) is built for each letter class. Chapter Chapter 5 discussed the different types of fuzzy models and the process of building each type and

its advantages and disadvantages. The second type, a table of features is built. It describes the shape of the letters, the position of the letter and their attachments. Section 3.2 describe these features and their extraction process.

The result of the character modeling stage is two features' databases. The first one contains the description of the fuzzy models of the letters. The second database contains the extra features of those letters.

6.1.2. Text Recognition

Text recognition is the second main phase in this work. The main input of this phase is the cursive text testing data. Two different databases are used to test the system. Section 6.3 discusses the details of these databases. In the preprocessing phase, the polygonal approximation is performed on each sample of the online text. Then the main body of each data sample is extracted, so the PSP detection process will not take the attachments into consideration. Sections 3.3 and 4.2 discussed the polygonal approximation and the process of extracting the text main body. The preprocessing phase is followed by two simultaneous processes, Text feature extraction and PSP Detection. Text features extraction aims mainly to extract the extra information about the text. It contains mainly the attachments properties, existence of loops, writing line information and other extra information. The features extracted in this process are used in the final process of text recognition enhancement. On the other hand the PSP detection is done to segment the preprocessed data. As described in chapter Chapter 4 the output of this process will be a set of segmented characters. These characters are modeled by generating their directions and lengths. Then a MTL are generated for each letter, section 5.2.2 describes the processes of generating these models and its importance. After

all preprocessing and modeling is concluded the letters are recognized using fuzzy comparison technique. The recognition of the online Arabic text is discussed in the following section. Figure 6.2 shows a detailed step by step algorithm of the text recognition phase.

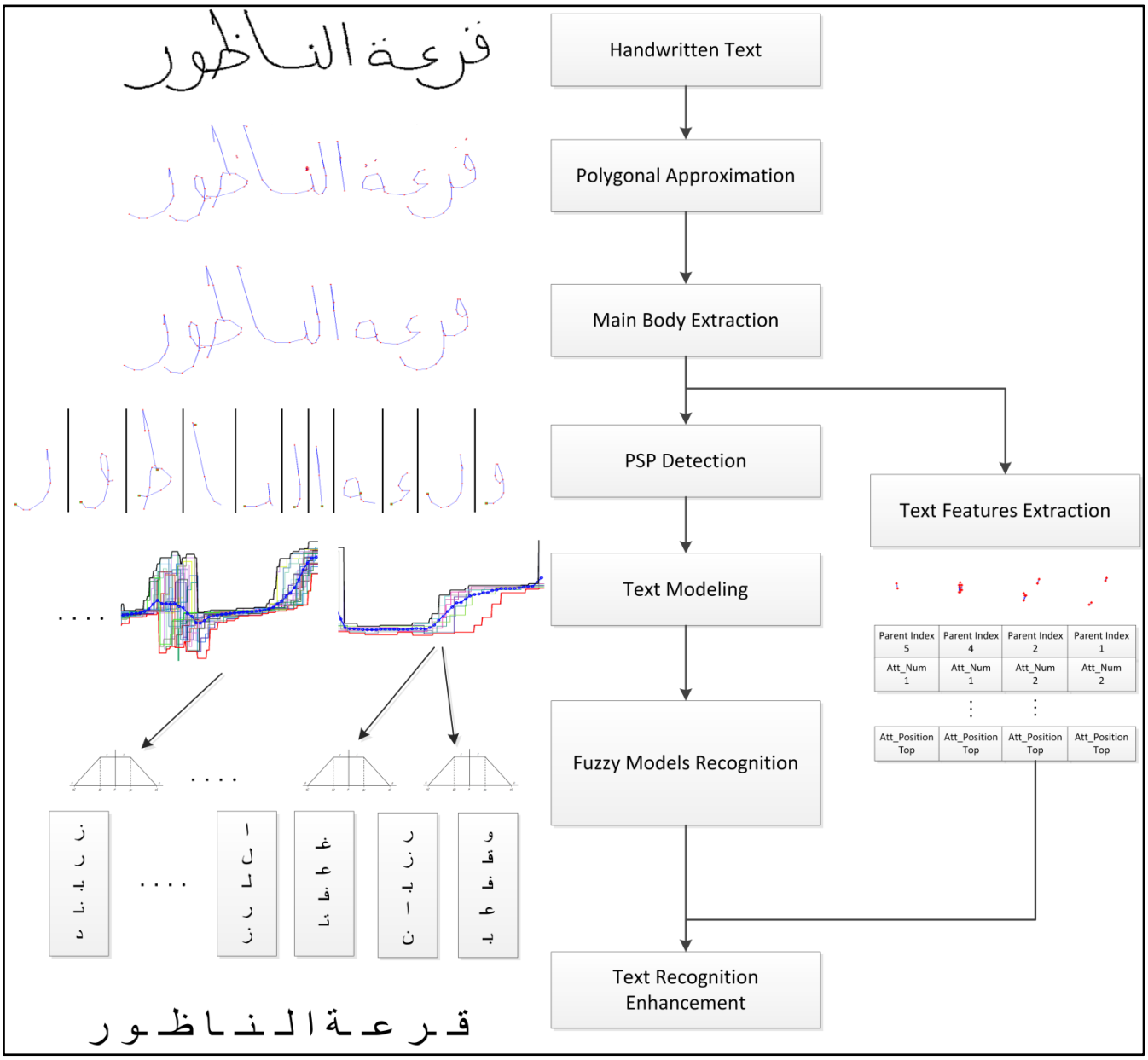


Figure 6.2: Illustration of Arabic handwritten text recognition.

6.2. Fuzzy Text Recognition

The general outline of Arabic handwriting recognition has been described in the previous section. This phase mainly depends on the fuzzy comparison of the developed fuzzy models. In this section we will illustrate how this fuzzy similarity between samples and trained models is applied.

In the training phase fuzzy models have been generated for the 54 models. Figure 6.3 shows 3 different examples of the MTL fuzzy models for different Arabic letters. In Figure 6.3-(a) the middle marked curve represents the MTL. The top and bottom curves represent the fuzzy tolerance envelops. The next section discusses this structure in details. In Figure 6.3-(b) the base writing shape of the letter/letters are showed. The participating letters in that model are listed in Figure 6.3-(c).

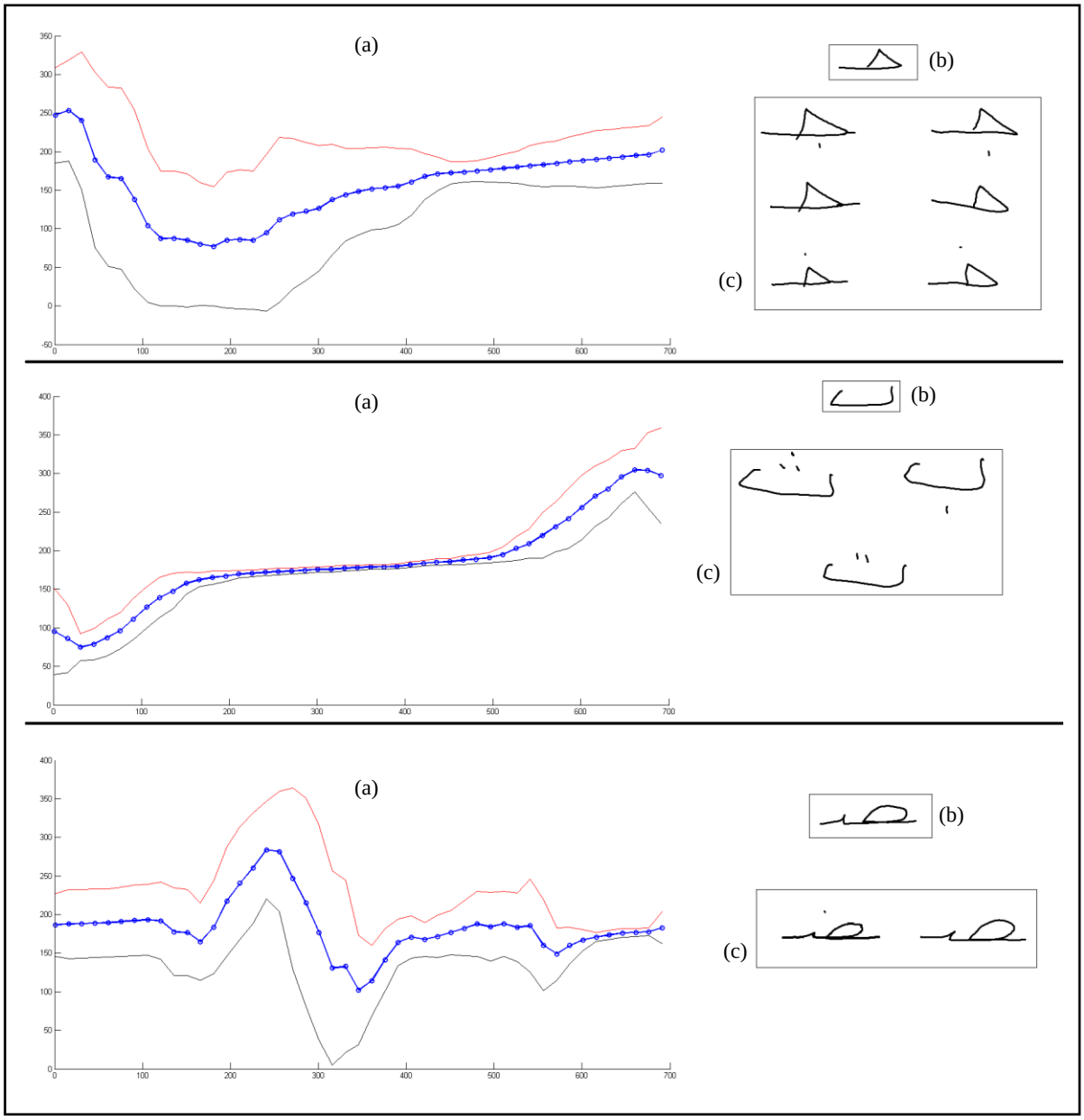


Figure 6.3: Examples of letters' fuzzy models. (a) The MTL fuzzy model. (b) The base shape of model writing. (c) The participating letters in the model.

In the fuzzy text recognition phase each data sample is compared to the fuzzy models. This comparison aims to find the most similar models to the sample; which implies that this sample is represented by one of the models.

6.2.1. Similarity Calculation

Each MTL is a curve of 47 sampling points resulting from a window of size 15 over the 700 points of the letter. Figure 6.4 shows two overlapped MTLs for a letter sample and model. All the sampling points are aligned where the sample curve is marked by triangular points and circular points for the model trend line.

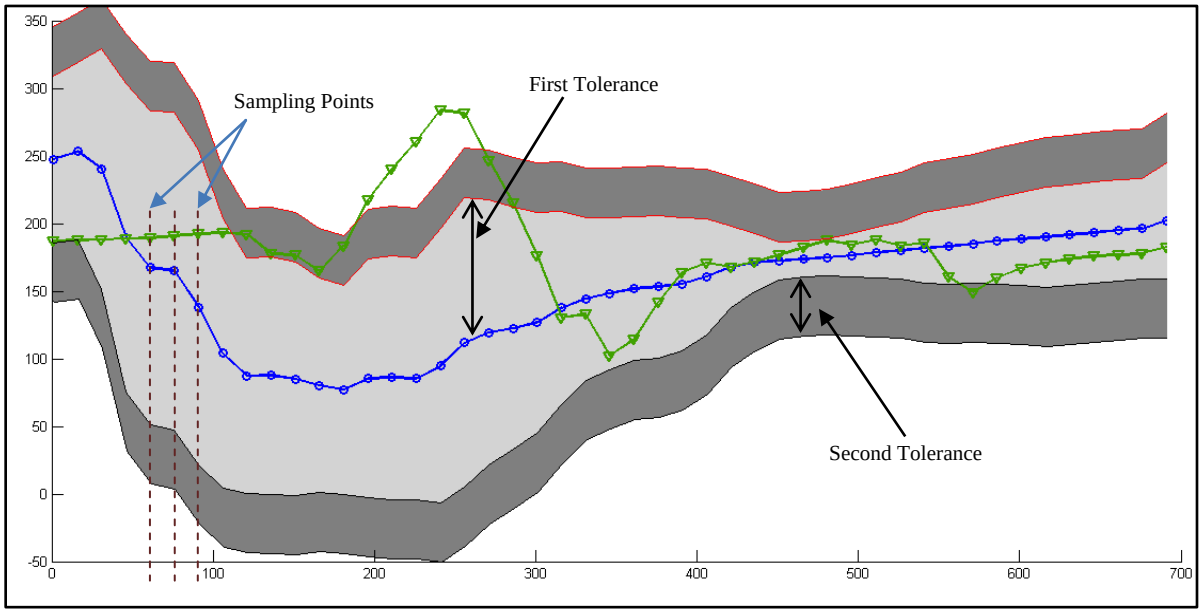


Figure 6.4: MTL of a letter sample aligned with a letter model

At each sampling point, fuzzy comparison between the sample and the model is done to estimate their similarity. The shaded area in Figure 6.4 represents the fuzzy tolerance of the model. The width of this area on both sides is related to the standard deviation which is estimated in the training phase. The width of the fuzzy tolerance varies from one point to another according to the model as estimated in the training phase. The membership of any point in the middle area is taken as 1 which represents the peak of a fuzzy set. The membership outside this area is calculated based on trapezoidal membership function as shown in Figure 6.5. When computing the membership value at a certain sampling point, the value of the membership will be determined based on one of the following four cases.

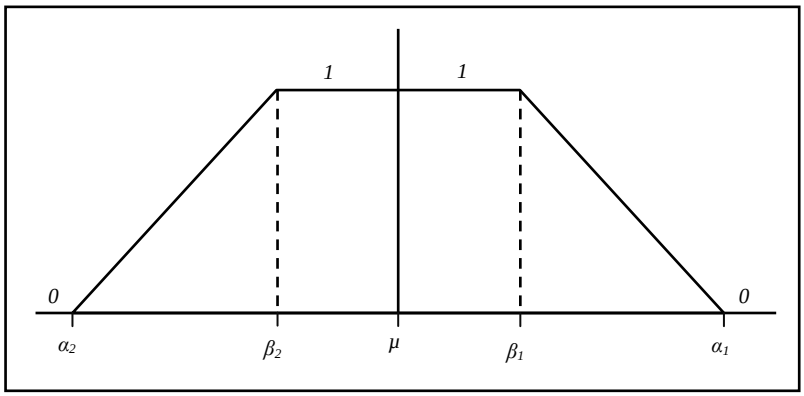


Figure 6.5: Trapezoidal membership function

The membership value of x is denoted by M_v.

$$Mv(x,m) = \begin{cases} 0, & x > \alpha_1 \quad or \quad x < \alpha_2 \qquad (Case\ 1) \\ 1, & \beta_2 \leq x \leq \beta_1 \qquad (Case\ 2) \\ \dfrac{\alpha_1 - x}{\alpha_1 - \beta_1}, & \beta_1 < x \leq \alpha_1 \qquad (Case\ 3) \\ \dfrac{x - \alpha_2}{\beta_2 - \alpha_2}, & \alpha_2 \leq x < \beta_2 \qquad (Case\ 4) \end{cases}$$

This membership value represents the weight of the similarity between the sample *x* and the model at that sampling point *m*. The value of $M_v \in [0, 1]$, such that 0 means least similarity between the sample (outside the tolerance region) and the model at that sampling point, and 1 means maximum similarity (inside the main tolerance region). Figure 6.6 shows these cases.

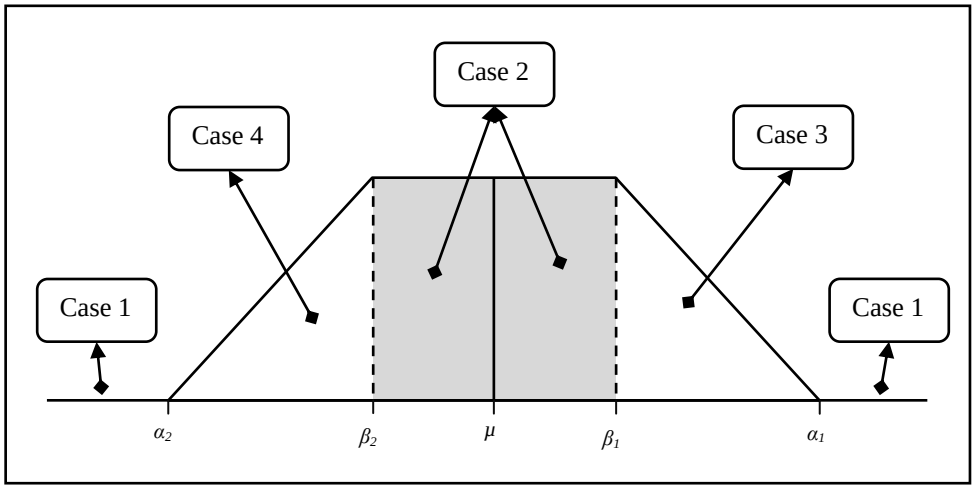


Figure 6.6: Membership cases

For letter sample *S* and a letter model *M* with I number of sampling intervals. The overall similarity between *M* and *S* will be.

$$sim(S, M) = \frac{1}{I}\sum_{v=1}^{I} M_v(S(v), M(v))$$

A similarity of value 1 between the sample and the model means that the sample is similar to that model. In other words, this model contains a letter that is similar to the recognized letter.

Figure 6.7 illustrates part of the fuzzy model in 3-dimentional represntation. This illustration shows the real form of the fuzzy model, including the two tolerance regions. The figure shows a number of sampling points in the model; the similarity check is done at each one of these points.

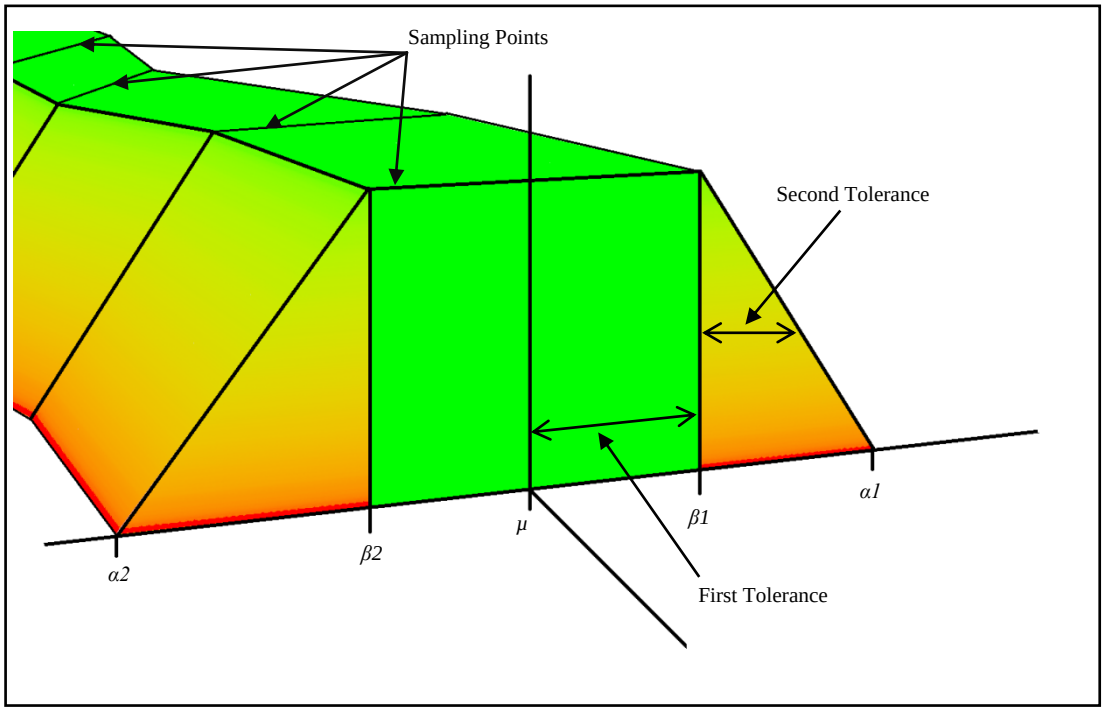


Figure 6.7: Part of a fuzzy model in 3-dimentional view

6.2.2. Directions and Lengths fuzziness

The computation of similarity using fuzzy turning function is introduced in [8]. In [8] the fuzzy comparison is applied using the turning functions of the samples and the models. It was limited to the fuzzy comparison of the direction values of both the sample and model. In our work we use the fuzzienss in both the direction and lengths. Figure 6.8 shows an example of a letter model with a letter sample (dashed curve).

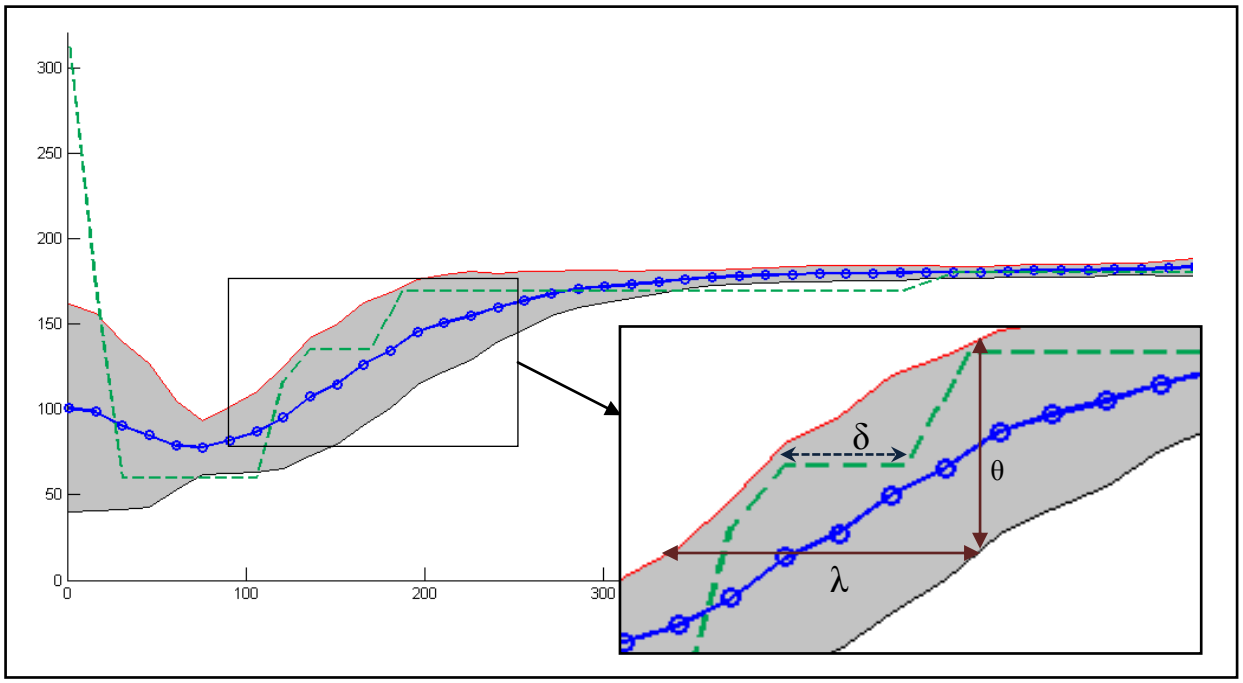


Figure 6.8: Matching of a model with a sample

We can see that the segments in the sample letter have different lengths and directions and still have a membership value of 1. This fuzziness is a result of building the models and taking the standard deviation at each sampling point in consideration. The fuzziness area (shaded area in Figure 6.8) gives the membership of value 1. We can notice in the zoomed part that the segment δ can have any length in the λ range and any direction in the θ range, and it still have a membership value of one.

This 2-way fuzziness in directions and lengths is powerful to recognize the letters with different styles of writing. In addition, it maintains the main common shape and structure of the letter. Figure 6.9 shows an example of 3 different samples written in different styles. The samples have one writing structure. All 3 samples fit the matching letter model. This illustrates how this model can tolerate different styles of writing and retains the main structure of the letter.

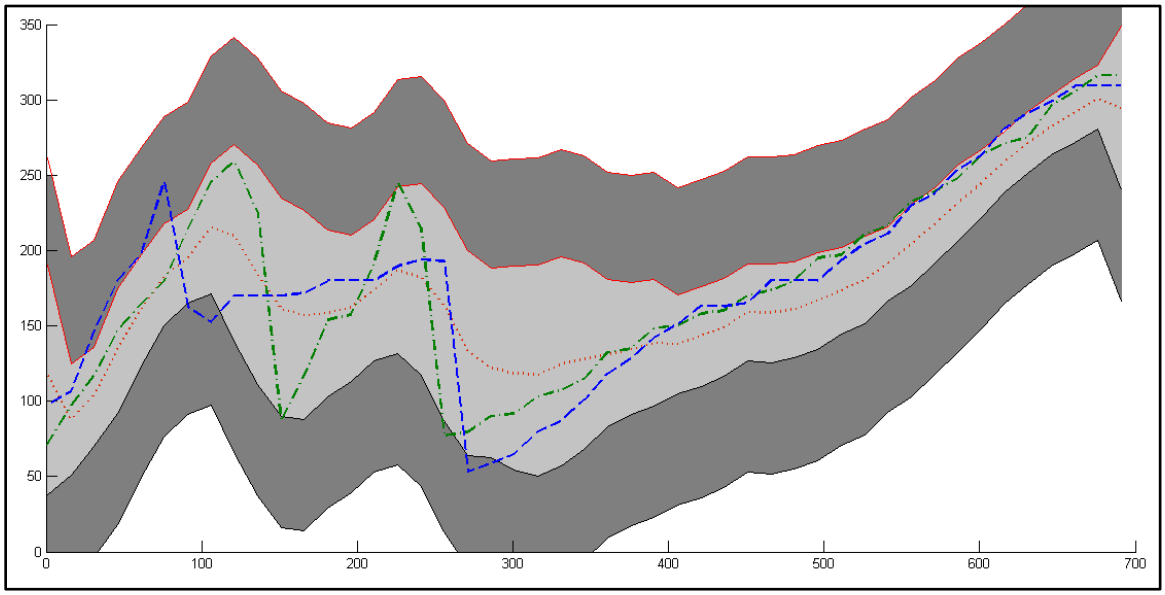


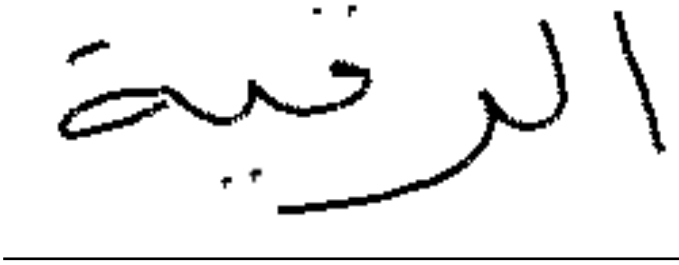
Figure 6.9: Different samples of 3 letters fitting one model

6.3. Experimental Results

In this section we present the experiments that were conducted to evaluate the Arabic text recognition algorithm. The details of the experiments and the used databases are discussed.

6.3.1. Recognition Measure

For a data sample of length *l* letters, the output of the recognition phase is a list of candidate letters at *l* positions. Each position represents a letter in the ground truth label of the sample. The recognition algorithm creates a candidate list of possible letters at each position. Each list contains the different letters (models) that can be considered as a match at that position. The letters in the lists are sorted based on its ranking. Figure 6.10 shows a data sample that contains the Arabic word (الرقبة). The word contains 6 letters, this means it has 6 possible letters' lists.



Pos6	Pos5	Pos4	Pos3	Pos2	Pos1	Position
ـة	ـبـ	قـ	ـر	ـلـ	ا	**Original Label**
ـة ة ق ط	ـبـ ـبـ ـهـ	ؤ ق فـ قـ	ـر ر ـدـ ن	ا ز ك ـلـ ـلـ	ا ل ر ـا	**Possible Letters**

Figure 6.10: Example of the recognition output

We used edit distance to evaluate the recognition rate of the results. It helps in calculating the number of modifications we need to match the expected letters with the ground truth label. Originally the edit distance algorithm is used to compare two strings. It calculates the addition, deletion and replacement costs to transfer one string into another. Two matching strings cost zero to be transformed. The edit distance can be used as a dissimilarity measure. In our work we modified the algorithm so we can compare one string with letters lists. The difference between the original edit distance algorithm and the modified one is that our algorithm uses (is-in-list?) operation instead of equality operation when comparing the letters. When comparing two letters we check if the original letter is in the possible letters' list. If yes it is considered as a matching position.

Figure 6.11 shows an example of the edit distance calculation between a label of sample (the word قرطاجة) and possible letters for that sample. The length of the ground truth is 6 letters, while the recognition system resulted in 5 possible letters' list. The edit distance output in the example is 3. This means that the recognized list needs 3 operations to match the output to the ground truth.

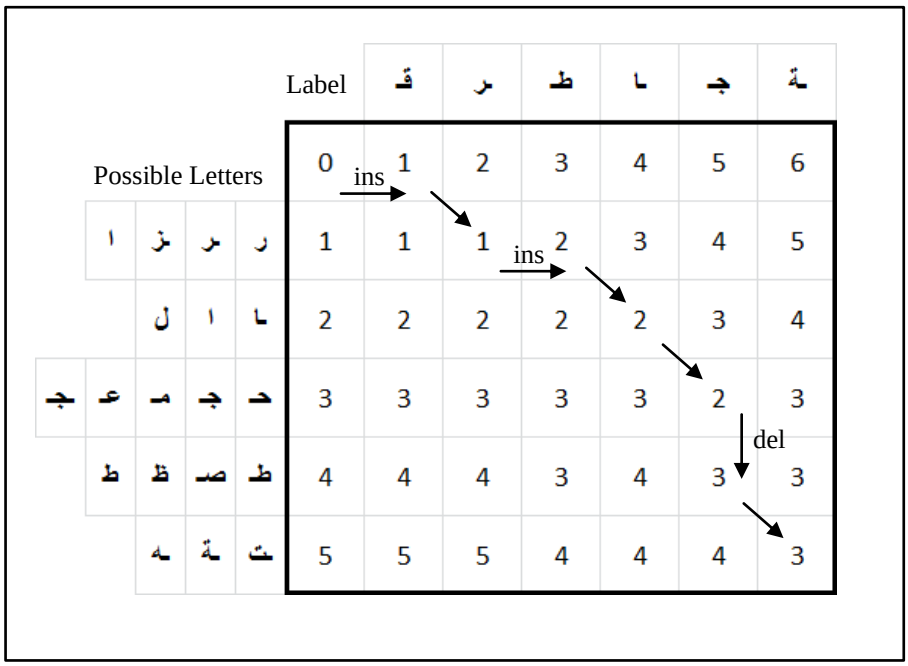


Figure 6.11: Edit distance matrix example

For a data sample label of length *l*, and possible letters of length *v* if the output of the edit distance matching is *d*. Then the similarity ratio is:

$$r = \frac{l - d}{l} \quad , 0 \leq r \leq 1$$

6.3.2. Arabic Online Text Databases

In our work, two types of data have been used to evaluate our work. The first one is our database; it is a small database of Arabic words. The database is characterized by its containment of all forms of Arabic letters. The database contains 193 samples each one contains one to three words. Figure 6.12 (a) shows some samples from the database.

The other database is called Online text dataset for Arabic DAtabase (ADAB) [34]. The database is composed of 33000 Arabic words written by more than 166 writers. The database is developed in cooperation between the Institute for Communications Technology (IfN), the Ecole Nationaled'Ingenieurs de Sfax (ENIS) and Research Group on Intelligent Machines (REGIM), Sfax, Tunisia. The database is composed of more than 900 Tunisian town/village names. The database is used in recent researches and has been used in a competition [35]. Figure 6.12 (b) shows some samples from the database. This database has some limitations as it is not generic database (its content is limited to Tunisian town/village names). It also has low quality of writing, where large numbers of the data samples are not readable by humans.

(a) (b)

Figure 6.12: Examples of data samples (a) Our database (b) ADAB database

6.3.3. Results and Discussion

We now present our results in the testing phase of our database and of the ADAB database. Table 10 shows the obtained results after performing all the tests. The ADAB database is divided into 3 sets we show the results of each set independently.

Table 10: Online Arabic text recognition results

		Rejection		Recognition Rates	
Database	**Database Size**	**Num of samples**	**Ratio %**	**Accuracy %**	**Correctness %**
Our Database	193	19	9.84%	43.84%	64.40%
ADAB Set1	5037	563	11.17%	24.44%	60.10%
ADAB Set2	5090	514	10.09%	25.59%	60.06%
ADAB Set3	5027	517	10.28%	29.5%	59.90%

A rejection process has been added to exclude the poorly segmented samples. Any sample that has PSPs more or less than the actual number of letters in the label is rejected. Hence, the increase in the number of PSPs in the sample should not exceed half the number of the letters in the ground truth label. The rejection of the samples is done based on the automatic evaluation of the PSP detection process.

The accuracy value represents the average of the similarity values of all samples. The similarity values calculation is described in section 6.3.1. The insertion, deletion and replacement cost is one. On the other hand, when calculating the correctness the deletion cost is not considered.

These results are in term of character recognition. We mainly work with separated letters, and the evaluation process is done on the character's level. Word recognition can be done as post processing phase. The recognized letters can be combined together, by using lexicon and dictionary to obtain the recognized words. The experimentation is done

without any use of any Arabic lexicon. However, it's expected that the use of dictionaries will improve the recognition rate. This part of work the will be conducted in the future.

As it's clear in the table, the highest correctness value obtained is 64.4%, which can be relatively considered competitive result in comparison to similar works on Arabic language using the structural features of the text.

CHAPTER 7

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this chapter, we summarize the contributions of this thesis. The results of the thesis are analyzed and discussed. This chapter also presents future research directions.

7.1. Conclusions

In this thesis, we addressed automatic online Arabic handwriting recognition. The online handwriting recognition has many applications like data input, forms processing, writer verifications and identification and many other applications. Different factors caused the lack of research on online Arabic text recognition. The cursive natures of Arabic writing, the existence of ligatures in addition to overlapping characters and lack of resources are the main factors.

Previous work on online Arabic handwriting recognition focused on isolated characters, numerals and words. Different techniques are used in the segmentation and recognition phases that mostly used statistical techniques. Classifiers such as Hidden Markov Models (HMM) and Neural Networks are used in the recognition phase. In this work we used structural methods. Structural techniques are not widely used in this field. Few systems applied a combination of statistical and structural techniques to perform the segmentation and recognition processes.

In the following we present the contributions of the thesis to online Arabic handwriting recognition using structural attributes.

- **Letters attachments:** A rule-based algorithm is applied that is easy to update. The rules validate the structural properties of the strokes in a data sample. The strokes that match the rules are considered as attachment. New rules can be added or current ones can be modified and removed easily. The algorithm also links each attachment to its most probable parent.
- **Possible Segmentation Points (PSP):** The proposed algorithm is rule based that uses a set of rules to identify the best PSP on the online text. The suggested rules are derived based on analyzing the connectivity of Arabic text and on the general features of text. The segmentation algorithm is evaluated manually and simi-automatically using our database and ADAB database.
- **Fuzzy structural models of Arabic characters:** The thesis proposed enhanced fuzzy models of Arabic letters. Polygonal approximations of Arabic characters are generated then a turning function model is generated from the polygonal approximation models. The general model of a letter is obtained by accumulating the set of samples of characters. This automatic model is used to generate a fuzzy model of Arabic characters with fuzzy directions and lengths. This model is considered as enhancement to previous work of Parvez [8] in the following:
    - We used fuzzy directions and lengths while only fuzzy directions are used in Parvez work.
    - We used adaptive fuzzy margins at the different parts of each model while fixed margins are used in Parvez work.
    - The models are generated automatically from the training samples.

This makes the model more robust and representative of the different writing styles of writers.

- **A recognition prototype of online Arabic handwriting text.** It utilizes all the proposed techniques in the thesis which provide multiple hypotheses resulting in list of possible letters. The lists are used to generate multiple hypothesized words, which can be used to improve recognition rates by using a language model or a dictionary.

Although the recognition rates are not ready for practical applications, this thesis resulted in a very effective fuzzy modeling of characters. In previous work, the fuzzy modeling was applied to the polygonal directions. In this work, we extended it to both directions and lengths. In the previous work the fuzzy direction was integrated in the similarity function, while in this thesis, we build real fuzzy models. The fuzzy function was improved from fixed intervals (main and side intervals) to flexible intervals generated automatically from the training data.

7.2. Future Research Directions

We consider that the main objectives of this thesis are met. However, many enhancements and extensions can be made to improve the recognition accuracy. In addition, the proposed techniques in this thesis can be extended to other fields and applications.

- **The building of letters' models can be extended to include different styles of writing**. This means that more than one model can be built for each class. These models can represent the different styles of writing of the letter.

- **Extension to include other types of attachments**. Our work addressed only the primitive type of attachments. So considering all types of attachments may increase the reliability of the recognition.
- **Using more adaptive version of the polygonal approximation**. The current version of polygonal approximation is occasionally over simplifying the online text. This causes the removal of some important corner points. Hence, the use of an improved algorithm is expected to improve the recognition rates.
- **Enhanced segmentation rules**. The current sets of segmentation rules need enhancement as it doesn't cover all the possible segmentation points. New rules may be added to assist in segmenting ligatures. Adding new rules may improve the segmentation accuracy and hence the recognition accuracy.
- **Enhance the process of building fuzzy models**. The process of fuzzy modeling can be enhanced by using clustering techniques. It may be used to generate more than one model from one set of training samples. This is expected to result in more robust models.
- **Models similarity calculation enhancement.** Using advanced statistical methods can improve the accuracy of similarity calculation. Possibly by using different weights for the models segments. Hence more narrow fuzzy parts of the letters' models are given higher weights. In addition, it can help in generating distinctive features' points in the letter model.
- **Using lexicon/dictionary and language model.** A post-processing phase may be added to generate possible multiple words. Then a language model with a dictionary can be used to select the best word that matches the context.

Acceptable recognition rates are achieved, given that it is applied on unconstrained online handwritten text. This includes the results of all phases. It is known that the segmentation errors increase the recognition errors nonlinearly. This result motivates us and other researchers as there is big room for improvements.

With this we conclude our thesis work. All prayers and thanks due to Allah for helping and permitting us to complete this thesis.

إن أصبت فمن الله و إن أخطأت فمن نفسي و الشيطان، و الحمد لله رب العالمين.

REFERENCES

[1] R. I. Elanwar, M. A. Rashwan, and S. A. Mashali, "Simultaneous Segmentation and Recognition of Arabic Characters in an Unconstrained On-Line Cursive Handwritten Document," *Engineering*, pp. 465–468, 2007.

[2] S. Al-Emami and M. Usher, "On-line recognition of handwritten Arabic characters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 704–710, Jul. 1990.

[3] S. D. Connell and A. K. Jain, "Template-based online character recognition," *Pattern Recognition*, vol. 34, no. August 1999, pp. 1–14, 2001.

[4] S. Kc and C. Nattee, "A Comprehensive Survey on On-line Handwriting Recognition Technology and its Real Application to the Nepalese Natural Handwriting," *Kathmandu University Journal of Science, Engineering and Technology*, vol. 5, no. 1, pp. 31–55, Mar. 2010.

[5] H. El Abed, M. Kherallah, V. Märgner, and A. M. Alimi, "On-line Arabic handwriting recognition competition," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 14, no. 1, pp. 15–23, Jul. 2010.

[6] S. Procházka, "'Arabic'," *Encyclopedia of Language and Linguistics*. 2006.

[7] E. Miller, Tracy, "Mapping the Global Muslim Population: A Report on the Size and Distribution of the World's Muslim Population" *Pew Research Center*, Octuber, 2009.

[8] M. T. Parvez, "Arabic Handwritten Text Recognition Using Structural and Syntactic Pattern Attributes," PhD thesis, King Fahd University of Petroleum and Minerals, 2010.

[9] Y. Assabie and J. Bigun, "Online Handwriting Recognition of Ethiopic Script," in *Eleventh International Conference on Frontiers in Handwriting Recognition (ICFHR2008)*, pp. 153–158, 2007.

[10] A. M. Zeki, "Segmentation Techniques for Online Arabic Handwriting Recognition : A Survey," in *Information and Communication Technology for the Muslim World (ICT4M)*, pp. D37 – D40, 2010.

[11] I. S. I. Abuhaiba, S. a. Mahmoud, and R. J. Green, "Recognition of handwritten cursive Arabic characters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 664–672, Jun. 1994.

[12] M. T. Parvez and S. A. Mahmoud, "Polygonal approximation of digital planar curves through adaptive optimizations," *Pattern Recognition Letters*, vol. 31, no. 13, pp. 1997–2005, 2010.

[13] A. T. Al-taani, "Recognition of On-line Arabic Handwritten Characters Using Structural Features," *Language*, vol. 1, pp. 23–37, 2010.

[14] Kam-Fai Chan and Dit-Yan Yeung. Recognizing on-line handwritten alphanumeric characters through flexible structural matching. Pattern Recognition, Vol 32, pp. 1099 - 1114, 1999.

[15] V. Kumar, "Template-based Writer-independent Online Character Recognition System using Elastic Matching," *Journal of Computer Science*, vol. 1, no. 1, pp. 15–18, 2011.

[16] M. Kherallah, S. Njah, a. M. Alimi, and N. Derbel, "Recognition of on-line handwritten digits by neural networks using circular and beta approaches," *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 164–169, 2002.

[17] N. D. M. Kherallah, A.M. Alimi, "On-line Recognition of Handwritten Digits by Self Organisation Maps Using Elliptic and Beta Representations," in *IEEE International Conference. SCS'04*, pp. 503–507, 2004.

[18] M. Kherallah, L. Haddad, A. M. Alimi, and A. Mitiche, "On-line recognition of handwritten digits based on trajectory and velocity modeling," *Pattern Recognition Letters*, vol. 29, no. 5, pp. 580–594, 2010.

[19] K. Assaleh, T. Shanableh, and H. Hajjaj, "Recognition of handwritten Arabic alphabet via hand motion tracking," *Journal of the Franklin Institute*, vol. 346, no. 2, pp. 175–189, 2009.

[20] B. Alsallakh and H. Safadi, "AraPen : an Arabic Online Handwriting Recognition System," in *Information and Communication Technologies, 2006. ICTTA '06*, pp. 1844 – 1849, 2006.

[21] M. S. El-Wakil and A. A. Shoukry, "On-Line Recognition of Handwritten Arabic Character Recognition," *Pattern Recognition*, vol. 22, no. 2, pp. 97–105, 1989.

[22] Jakob Sternby, Jonas Morwing, Jonas Andersson, Christer Friberg, On-line Arabic handwriting recognition with templates, Pattern Recognition, Volume 42, Issue 12, Pages 3278-3286, December 2009.

[23] T. S. Al-Sheikh and S. G. El-Taweel, "Real-Time Arabic Handwritten Character Recognition," *Pattern Recognition*, vol. 23, no. 12, pp. 1323–1332, 1990.

[24] M. A. H. Omer, “Online Arabic handwriting character recognition using matching algorithm,” *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, pp. 259–262, Feb. 2010.

[25] A. M. Alimi, “An Evolutionary Neuro-Fuzzy Approach to Recognize On-Line Arabic Handwritting,” in *Proc. Int’l Conf. Document Analysis and Recognition*, pp. 382–386, 1997.

[26] a. M. Alimi, “A neuro-fuzzy approach to recognize Arabic handwritten characters,” *Proceedings of International Conference on Neural Networks (ICNN’97)*, vol. 3, pp. 1397–1400, 1997.

[27] F. Bouslama and A. Amin, “Pen-Based Recognition System of Arabic Character Utilizing Structural and Fuzzy Techniques,” in *Proc. Second Int’l Conf. Knowledge-Based Intelligent Electronic Systems*, no. April, pp. 76–85, 1998.

[28] N. Mezghani, M. Cheriet, and a. Mitiche, “Combination of pruned Kohonen maps for on-line arabic characters recognition,” *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, vol. 1, pp. 900–904, 2003.

[29] B. Fadi, E. Jihad, and H. Nizar, “Online Arabic Handwriting Recognition Using Hidden Markov Models,” in *The 10th International Workshop on Frontiers of Handwriting Recognition*, 2006.

[30] G. Al-Habian and K. Assaleh, “Online Arabic handwriting recognition using continuous Gaussian mixture HMMS,” *2007 International Conference on Intelligent and Advanced Systems*, pp. 1183–1186, Nov. 2007.

[31] K. Daifallah, N. Zarka, and H. Jamous, “Recognition-Based Segmentation Algorithm for On-Line Arabic Handwriting,” *2009 10th International Conference on Document Analysis and Recognition*, pp. 886–890, 2009.

[32] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell, “An efficiently computable metric for comparing polygonal shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 209–216, Mar. 1991.

[33] W. S. Cleveland, “Robust Locally Weighted Regression and Smoothing Scatterplots,” *Journal of the American Statistical Association*, vol. 74, no. 368, pp. 829–836, 2012.

[34] H. El Abed, M. Volker, M. Kherallah, and A. M. Alimi, “ICDAR 2009 Online Arabic Handwriting Recognition Competition,” in *International Conference on Document Analysis and Recognition*, 2009.

[35] H. El, A. Monji, K. Volker, and A. M. Alimi, “On-line Arabic handwriting recognition competition ADAB database and participating systems,” International Conference on Document Analysis and Recognition (ICDAR) , pp. 1454–1458, 2011.

VITAE

Name : Khaldoun ‘Mohammad Khaled’ Hassan Halawani

Nationality : Palestinian

Date of Birth : June 22 1987

Email : Kldoon@live.com

Address : Shabba – Hebron – Palestine

Mobile : +970598311741

Academic Background : BSc in Information Technology from Palestine Polytechnic University in 2009